TURBO FREEZER XL/XE 2011

Manual for software version 3.10
Last updated on March 30, 2013

Contents

Preface

Preface by the designer of the new freezer (2012)

TURBO FREEZER 2005 was a big success. It was very well received in the Atari community and we got lots of positive feedback and ideas for new features. The decision to use a flash ROM instead of an EPROM proved being particularly helpful. It enabled users to enjoy software updates and new features as the flash ROM could be updated from the Atari.

In particular the debugger has been extended considerably since the first release 2005. An online help, commands for moving and comparing memory blocks, a memory usage map and an OS vector overview, the possibility to log the screen output of debugging sessions to printer in parallel and many additional things have been added. Special thanks go to Erhard Pütz and Carsten Strotmann for many good ideas.

The initial run sold out fairly quickly and requests for more kept coming. In 2010 things became more concrete. Together with Wolfram Fischer, who also participated in the first run, I started planning a new production run. Meanwhile the Lattice M4A5 CPLD had become relatively hard to get and was clearly more expensive than comparable chips. We also wanted to use a slightly "larger" chip in order to implement some new ideas and features. So we decided to use a Xilinx XC95144XL as the basis for the new freezer. Having about the same price it offered more than twice the power (I/O pins and space for logic) than the Lattice CPLD. This way we had the possibility of implementing long-awaited features – for example the RAM extension which had been there in the very first freezer from 1987.

By the end of 2010 Wolfram had created the first prototype of the new hardware. This was the basis for starting the development of the new logic and software at the begin of 2011. I realized quickly that the best way would be to develop the logic from scratch and to separate the functional blocks (freezer, cartridge emulation, etc.). This simplified extensions in each functional block.

The logic of the cartridge emulation was completely redone; the freezer logic has intentionally been kept compatible with old TURBO FREEZER 2005 logic. I wanted to keep the changes to the freezer software at a minimum, so it can also be used with the "old" freezer. This way the users of the old freezer may also benefit from further updates.

In summer 2011, after several new hardware revisions and many changes to the software and the logic, the new freezer was completed! Unfortunately the start of the mass production, which had been planned for end of 2011, had to be postponed by almost one year. Wolfram had to move to a new residence and was busy at his job – hence we both had almost no time left for our hobby. Thanks to the support from the ABBUC the production finally started in autumn 2012! Many thanks to Wolfgang Burger for the support. Also

thanks to all who ordered the new freezer in advance and waited so patiently.

Salzburg, November 2012

Matthias Reichl

P.S. (March 2013): I'd like to add a big "thank you" to Peter Dell who took over the job of translating the German manual into English. At the end this was a whole lot more of work than expected, but Peter was really brave and did an awesome job – thanks Peter!

Preface by the designer of the new freezer (2005)

The project to design a new TURBO FREEZER started in early 2004, when Bernhard Engl gave permission to the ABBUC to remake his products (The TURBO 1050 and the TURBO FREEZER). To me, the TURBO FREEZER has always been the most interesting piece of hardware extension that has ever been available for the Atari and before starting the project, I had already studied and analyzed how it worked. That's why it was pretty obvious that I would engage in rebuilding the TURBO FREEZER. There were more participants in the team: Bernhard Pahl, Florian Dingler, Torsten Schall, Guus Assmann, Frank Schröder and towards the end also Wolfram Fischer.

So I would like to express big thanks to all of you. The project wouldn't have come this far without you!! And some special thanks to Guus, for the many ideas for extensions, designing the PCBs and producing the prototypes. Thanks as well to Bernhard Engl for his tips and the detailed information for the original TURBO FREEZER. All other team members, thanks for the support and the testing!

My original plan was to remake the freezer, using modern parts. At this point, I never imagined that the result would be such extensive and powerful improvements! During the project, both hardware and software have been gradually extended and improved up to a point where only the basic (and genius) principle idea of the freezer was left. Everything else had been changed.

Some of the extensions came by as pure coincidence. For example the cartridge emulation: As 16K EPROMs were much more expensive, Guus suggested using a flash ROM. To enable programming this flash ROM from the Atari, it was needed to map the chip into the memory-map of the Atari. This formed the basis for the cartridge emulation :-)

Likewise with the RAM: The original 2K RAM was hard to get and also quite expensive. So the first step was to upgrade to a 32K chip and also use 128K flash ROM. The next step was to go to 128K RAM and this made it possible to put a memory-snapshot into this RAM. And the flash ROM size was increased to 512K. To enable this, the logic had to be improved. And a bigger logic chip gave space for the emulation of the 16K OSS modules and the SpartaDOS X module.

With the current state of the logic chip's internals, there's no space left and we're all content to have gotten the maximum out of the parts that were used. For the software, there are some more ideas for extensions, but that's all I'll say for now.

During the development process, I've constantly kept in mind that I wanted to create an open basis for further developments. For this reason, all design information is freely available and there's even a JTAG interface on board so the logic can be reprogrammed.

This will help anyone who wants to expand the freezer or even make something completely new with it.

All relevant information and updates for the TURBO FREEZER XL/XE 2005 are available on the internet at `http://turbofreezer.horus.com/`. There's also a manual on how to alter the logic of the TURBO FREEZER. For technical questions, write an e-mail to `hias@horus.com`.

The many features of the TURBO FREEZER have not only convinced the team members, but also many members of the ABBUC. In their Hardware Contest 2005, it was awarded a very convincing first prize.

I wish you a lot of fun using the TURBO FREEZER and stay loyal to the Atari 8-bit computers for many years to come.

Salzburg, December 2005

Matthias Reichl.

Preface by the developer of the original freezer (2005)

As a designer, the best recognition of achievement one can get, is that a once successful product is continued and enhanced by a competent successor. However, if this happens after 16 years and for a home-computer product, it's downright sensational and means that the product has attained cult status. While using original concept of the 1980's, Matthias Reichl has made a real masterpiece of the TURBO FREEZER XL/XE 2005. The application of modern micro-chips has led to a great extension of the functionality. He even managed to remove some minor bugs that were known, but couldn't be prevented using the electronics of that time, without making the product economically impossible because of the higher chip-count.

The improvements and enhancement, especially the ingenious cartridge emulation that was not present in the original, make the TURBO FREEZER XL/XE not only the best and most perfect freezer for the 8-bit Atari that ever existed, but also is an all-rounder talent, having the potential to become a legendary cult product. At this point in time, it's a high-light of a technical development that started more than 20 years ago. In my notebook, which has turned yellow now, the first sketches of the concept date back to the autumn of 1984. Finding this out even surprised myself.

Experiencing a vastly improved, enhanced and rejuvenated new series of a computer extension product, is not only a miracle, but also a special joy and honor to me, so I wish all proud owners of the new TURBO FREEZER XL/XE a lot of pleasure using it.

Zurich, November 2005

Bernhard Engl

Preface by the developer of the original freezer (1987)

After one year development, hundreds of used-up integrated circuits, 5300 lines of assembler code and five-digit costs, it's there at last: The TURBO FREEZER XL. It's not only

the first real freezer for the Atari, it is also the only extension a freak needs to get the optimum and maximum usage out of his Atari. This is because apart from freezer, the board also offers a socket for an Oldrunner and for 256k of RAM so an 800 XL can be extended to up to 320k.

By using three ASICs containing the equivalent of over 40 TTL circuits and by avoiding a costly multi-layer PCB, all this could be implemented at an unbeatable price. In addition, thanks to the very powerful memory management that is anyway needed for the freezer, the computer doesn't need to be opened.

The careful same tuning of all components can be found in the built-in software which consists of the freezer, a mini-DOS and a debugger. Combined with the hardware, this power-tool gives an unprecedented possibility of control over programs to the user. That's why anyone who's seen the TURBO FREEZER XL in action, wanted to have one as well. The thrill of having total control over the computer, even though the programmer did not want this, is well worth the price. And then there's also the money save from buying separate extensions that would be needed to get the same functions that are present in the TURBO FREEZER XL.

More about this in the description of the individual functions! I wish every proud owner lots of fun with the most fascinating product that was ever available for the Atari.

Munich, May 1987

Bernhard Engl

1 Installation

The TURBO FREEZER is simply attached to the parallel bus. The parallel bus differs between Atari XL and XE computers. In Atari XL computers the parallel bus consists of the Parallel Bus Interface (PBI) which is labeled “Parallel Bus”. In Atari XE computers the parallel bus consists of the cartridge port and the Enhanced Cartridge Interface (ECI) which are labeled “Cartridge” and “Expansion”. The power supply of the TURBO FREEZER is taken from the parallel bus. In Atari 600 XL and older Atari 800 XL computers the required power is available directly on the parallel bus. It is no longer available there in new Atari 800 XL models and therefore has to be taken from the joystick port using the power supply wire that is soldered to the freezer. Neither opening the Atari nor soldering of any kind is required. Yet users with experience in soldering and with the appropriate equipment have the option to add additional features.

The TURBO FREEZER consists of two parts, the freezer electronic in the black case and an adapter board which establishes the connection to the PBI or ECI. The adapter boards are available separately and hence provide a simple and cheap way to use a TURBO FREEZER with Atari XL as well as with Atari XE computers.

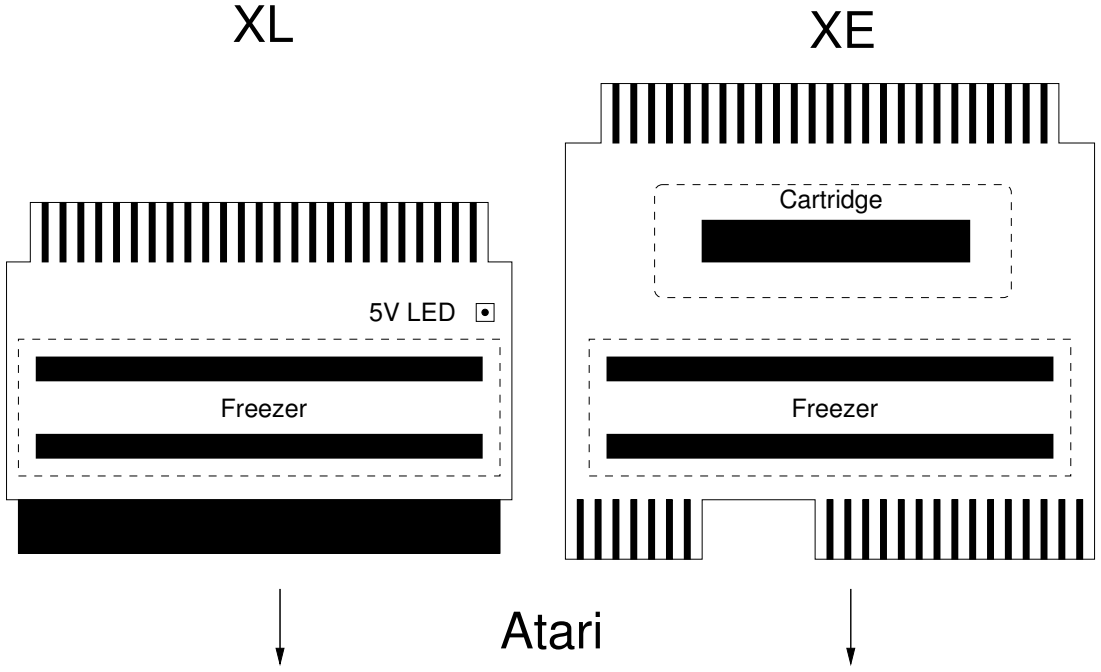


Figure 1.1: Atari XL and XE adapter boards

There is a small number of models from the XE series (mainly 65 XE) which do not have an ECI port. These models only have a “Cartridge” port and have no “Expansion”

port. The freezer can unfortunately not be used with these XE models.

Caution: The freezer electronic and the adapter board are not connected when they are shipped to prevent damage during transport. The Atari should always be switched off to also prevent damage when attaching or detaching the adapter board to/from the Atari or when attaching or detaching the freezer electronic module to/from the adapter board.

1.1 Verifying the power supply

Owners of Atari XL computer should verify the availability of the 5V power supply on the PBI before assembling the freezer. Owners of Atari XE computer can skip this step.

First switch the Atari off. Then attach the empty Atari XL adapter board without the freezer electronic module to the PBI. Now switch the Atari on. If the 5V LED on the adapter board lights up, then the 5V power supply is available on the PBI and the power supply wire to the joystick port is not required. Even so, the external power supply wire should not be cut off, but rolled up and put away because it may be required for use on a different Atari without the PBI power present. After that, switch the Atari off again and detach the adapter board from the PBI.

1.2 Assembling the freezer

1.2.1 Protecting from electro-static discharge

You should **electrically ground** yourself before assembling the freezer to discharge electro-static potentials which may be caused for example by the carpet, shoes, clothing and so on. These potentials can reach several 1000 volts and might destroy the XILINX chip and other ICs of the freezer. **Also never touch the multi-pin connectors of the freezer without being grounded.** This is true in general for all kinds of electronic parts. And though the Atari is not too sensitive in this regard, it may still get damaged.

You best touch the heating or the protective earthing conductor pin of a power outlet. Well suited, and very common in the professional area, are ESD (electro-static discharge) wristbands to ground you permanently.

1.2.2 Connecting the freezer electronic with the adapter board

At the bottom of the module with the freezer electronic there are two multi-pin connectors with 2x25 pins each. On the adapter board there are two female connectors with 2x25 pins each. The freezer module with the freezer electronic must be plugged into the adapter board in a way that the front side with the switches points towards the PBI or ECI port of the Atari. **The module must never be plugged in the other way around because that may destroy the freezer.**

At first the module with the freezer electronic must be positioned over the female connectors in a way that all pins of the multi-pin connectors are located over the corresponding female connectors. **No pins must overlap.** The best way is to hold the module with the

electronic in one hand and the adapter board in the other hand. Then place the module carefully on the board but do not press it and check from all sides that the pins are aligned correctly. Now connect the module with the adapter board using firm pressure. This works best if you turn the freezer upside down. Use the fingers of both hands to hold the module on the left and right side and then press the adapter board firmly in from the bottom with your thumbs.

After plugging the module in you should verify that all pins are completely inserted in the female connectors. If there are pins overlapping on one side, the module with the freezer electronic must be unplugged and must be plugged in again with correct alignment. If the pins are not completely inserted into the female connectors you should firmly press until no part of the pins can be seen anymore.

1.2.3 Disconnecting the freezer electronic from the adapter board

First the Atari must be switched off and the freezer must be detached from the PBI or ECI port. The freezer electronic and the adapter board are quite tightly connected via the 50 pin connectors. **You must never use too much force or perform abrupt moves** otherwise the pins may break and the freezer can be destroyed.

The following procedure works best: Lift the freezer module with the thumb by pushing the left and the right edge of the freezer module in the middle between the two multi-pin connectors. Lift it step by step carefully by 0.5 up to at most 1 mm on one side and then on the other side. Make sure to pull out the contacts equally on the left and right edge as well as in the front and the back row.

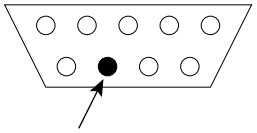
After about 2 mm the female connectors lose their grip. Now you have to proceed with special precaution because otherwise the pins will be bent easily. After about 5 mm you are done and the freezer electronic is disconnected from the adapter board.

1.3 Plugging the freezer in

1.3.1 Atari 600 XL (at least 64k RAM) and Atari 800 XL

Remove the snapped in PBI plastic cover, bend the metal shielding carefully away a little bit with your fingers and attach the freezer without using force.

If the PBI has a 5V power supply (see section 1.1), the connection to the joystick port can be omitted. If the power supply is not available on the PBI, the power supply wire must be connected to pin 7 of the joystick port 2. Pin 7 is marked in the figure on the right.



Caution: The power supply wire must never be disconnected while the Atari is switched on, otherwise the Atari and/or the freezer can be damaged. If the power supply wire is disconnected accidentally, the Atari must be switched off immediately.

1.3.2 Atari 800 XE, 65 XE and 130 XE

Attach the freezer to the cartridge port and the ECI port at the back of the Atari.

1.4 Turning the freezer on

1.4.1 Setting the basic configuration

Turn all freezer switches to the left to activate the following basic configuration:

- "**CartEmu**": `OFF`, cartridge emulation off
- "**FlashWrite**": `OFF`, write enable for flash ROM off
- "**OldOS**": `OFF`, Oldrunner off
- "**Stereo**": `OFF`, stereo POKEY mode off (not present)
- "**Ramdisk**": `OFF`, 512k RAM extension off

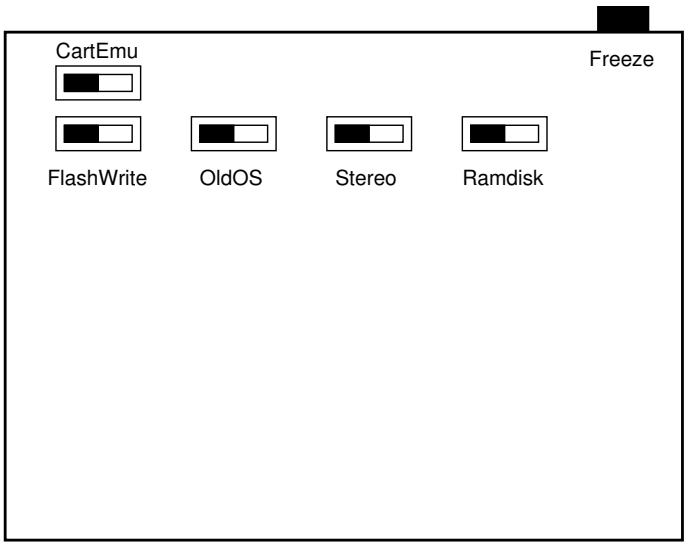


Figure 1.2: Switches on the TURBO FREEZER

1.4.2 Activating the freezer

Switch the Atari computer on and leave all other hardware switched off. If the `READY` message of the BASIC does not appear after the usual short delay, but other suspicious symptoms occur (system crash, black screen, smoke), switch the system off immediately and look for the problem. If the Oldrunner is active, the message `ATARI COMPUTER - MEMO PAD` is displayed instead of the `READY` message. If the cartridge emulation is active, the green screen with the `TURBO CARTRIDGE` menu of the cartridge emulation is displayed. In these cases verify that you have really setup the basic configuration according to section 1.4.1.

If READY is displayed, simply press the "**Freeze**" button on the upper right to display the freezer main menu. If nothing happens, the freezer is attached poorly, the power supply wire (if present) is not connected correctly or the freezer broke down. By pressing the [SPACE] key you can return from the freezer main menu to the BASIC. Now verify if BASIC reacts to keyboard input. If this is not the case, then there is a problem. If everything worked fine up to this point the freezer is ready.

If the freezer does not work correctly for any reason, send a letter or e-mail with the description of the symptoms first before sending the freezer back. Because of the 100% final assembly tests of every freezer it is practically impossible that the problem is due to the freezer. Most likely the assembly or the Atari itself are the cause for the problem. It is better to rule out these possible reasons before sending the freezer back for verification. This prevents unnecessary delays and costs.

1.4.3 Creating the system disk

Right after the successful installation of the TURBO FREEZER a system disk should be created. The system disk contains the flash program for writing to the flash ROM from the Atari and the freezer software. With this disk the TURBO FREEZER can be reset at any time to its state at delivery. In the state at delivery the flash ROM contains the software required for creating the system disk as follows:

1. Switch the Atari off and connect a disk drive.

2. Turn the "**CartEmu**" switch right to activate the cartridge emulation.

3. After switching the Atari on the menu of the cartridge emulation should appear.

4. Press [D] to load the default settings and confirm them with [RETURN].

5. Now you are in the "TURBO FREEZER System Disk Writer" software. Insert an empty disk in the disk drive "D1:" and press [RETURN]. The software now formats the disk in medium density and writes the freezer software. If an error occurs (e. g. due to a disk defect), the message ERROR is displayed and you have to insert a new disk and restart the program. To this end simply confirm the `restart program?` prompt with [Y]. To be on the safe side, a second copy of the system disk should be created. Write protect this second copy and store it in a safe location.

6. After creating the system disk successfully, switch the Atari off and turn the "**CartEmu**" switch back (to the left) again to its original position.

If anything goes wrong and you lose the system disk or overwrite it accidentally (yes, things like that can happen sometimes), there still is the option to download an ATR disk image of the system disk from the internet at `http://turbofreezer.horus.com`.

1.4.4 Activating the stereo POKEY mode (optional)

If the Atari has a stereo POKEY extension that is in stereo mode, the "**Stereo**" switch must be turned right. This way the freezer also saves the hardware registers of the second POKEY in the shadow RAM. When the freezer is activated, the second POKEY is disabled. When the program is resumed, its correct previous state is restored again.

The general rule is that the position of the switch must correspond exactly to the configuration of the Atari. Changing the switch position during operation is allowed, but then the shadow RAM will contain wrong information. To fix this you have to press `[RESET]` to trigger a re-initialization of the custom chip area by the operating system. Alternatively change the switch position only when the Atari is switched off.

1.4.5 Activating the integrated 512k RAM extension (optional)

As a special feature the freezer contains an integrated, battery backed 512k RAM extension. With this every Atari can be extended easier than ever from 64k to a total of 576k. And since the RAM extension is battery backed, the data in the RAM extension remains intact also after switching the Atari off.

Turning the "**Ramdisk**" switch right activates the RAM extension. A possibly present internal RAM extension in the Atari is automatically deactivated then.

1.4.6 Using cartridges with the Atari XE

Because the freezer occupies the cartridge port on Atari XE computers, there is an additional cartridge port present on the XE adapter board. With this you can continue to use cartridges even though the freezer is attached. The cartridges have to be plugged in the adapter board with the label towards the Atari. Never plug the cartridge in the wrong way, otherwise the cartridge, the freezer and the Atari can be damaged.

There are cutouts around the cartridge port where a "plastic tray", like those that are present in the cartridge port of Atari XL computers, can be inserted. This relieves you from the tedious unlocking some cartridges require. These "plastic trays" are unfortunately not available separately. The only way to obtain one is taking a defective Atari XL computer apart.

1.5 Extended configurations for experts in soldering

Warning: All of the following extensions are definitely only for experts in soldering. If you are not such an expert strongly consider finding one rather than trying this yourself. It is not worth ruining the poor Atari or the freezer due to unhandiness.

1.5.1 Providing an internal power supply

Whoever owns an Atari 800 XL without power supply on the PBI can add this retroactively, but this is difficult and requires good skills. If you are not confident enough you can simply

prolong the power supply wire of the freezer to the power supply inside the Atari. In both cases the joystick port remains free for its actual purpose.

To discourage beginners, the following description of the modification is limited to the absolute minimum. Experts will get along without problems. If you found an "expert" and realize that he is groping in the dark, then this is the last chance to stop this obvious amateur.

You can tap the power supply at the end of L1 from where a conductor path leads to the shielded part. The ferrite bead is located within spitting distance to the power switch. To this spot you can either solder the prolonged power supply wire of the freezer, or use two wires to connect the 5V power to pins 47 and 48 of the PBI. These pins are the second to last pins before pin 49 and 50. One is one the upper side of the board, one is on the lower side of the board. Soldering to the lower side is much more difficult because the Atari must be disassembled completely for this. In addition soldering to the ends of the mating surface without having solder creeping to the active part of the mating surface is finicky. The latter must be prevented in any case.

1.5.2 Installing a SYSTEM RESET key

For using the Oldrunner and for freezing programs which do not use interrupts, it is beneficial to install a `[SYSTEM RESET]` key which triggers a non-maskable interrupt (NMI) that cannot be disabled. The key button is installed such that pin 6 of the ANTIC can be switched to ground. It is best to connect the key button to the corresponding pull-up resistor. You can tap the pin at best at the pull-up resistor. In the Atari 800 XL and all Atari XE models this resistor is R31. In the Atari 600 XL it is R34, and in the Atari 1200 XL it is R7.

Warning: In addition to the general warning from the first section it applies here that mechanical works are required to install the key button. These works require additional tools and manual skills to achieve a good-looking result.

1.5.3 Providing compatibility with 1MB RAM extensions

The TURBO FREEZER uses the refresh line to stop the Atari. Unfortunately this leads to compatibility problems with all internal RAM extensions that use an own refresh logic based on the refresh signal of the ANTIC. This applies mainly to 1MB RAM extensions such as the Newell RAM extension. Most of the 256k RAM extensions and the extended memory of the Atari 130 XE are not affected. The problem can be solved easily with two additional components. You require a small signal Schottky diode, e. g. of type "BAT 85" and a 4.7kOhm resistor.

Warning: In addition to the general warning from the first section it applies here that the pin of the ANTIC must be treated with special precaution. The pin breaks very easily if it is bent too much.

After opening the Atari pin 8 of the ANTIC must be bent up. The wire that leads from pin 8 to the RAM extension must be unsoldered first. If the ANTIC is socketed, pull the ANTIC out of the socket and bend the pin up carefully. It is sufficient to bend the pin up only far enough that it stays outside of the socket when the ANTIC is put back into the

socket. If the ANTIC is soldered to the board, cut the pin right above the board with a mini wire cutter and bend it up carefully.

At first solder the diode between the bent up pin 8 and pin 8 of the socket (or the Atari main board if there is no socket). The cathode (marked with a ring on the housing) must be connected to the ANTIC; the anode must be connected to the socket. The most simple solution for this is to solder the diode directly to pin 8 of the ANTIC and to solder a thin wire from the anode to the lower side of the board and connect it there to pin 8.

One end of the resistor must be connected to the anode of the diode; the other end of the resistor must be connected to pin 21 of the ANTIC (+5V). Now the wire that was unsoldered from pin 8 of the ANTIC can be soldered again to pin 8 of the ANTIC.

2 Usage

Did you ever feel annoyed by a game that is really great but has no pause function that can be activated at any time? Or by games of the frustrating kind where you have to start all over again at the beginning after you lost all lives and then have to play forever to get to the point that you had already reached before? Or those games where the action only starts in the higher levels but where you have to waste huge amounts of time reaching these levels? In this case the freezer is exactly the right thing for you: a program can be frozen at any time in any situation and can be saved to any mass storage. From there it can be loaded at a later point in time and can be resumed at exactly the point where it was frozen. And this can be repeated as often as required, so it is no problem to spend a hundred lives to master a challenge though you only have one or two lives left.

In order to really be fun, a freezer must be available at any time, must have resident software (so it can work without cumbersome loading from disk or cassette) and must work fully automatic and within seconds. Therefore no cost cutting measures, like for example partly reconstruction of the hardware registers by user input or loading the freezer software from disk, were applied. Instead of applying such dubious cost cutting measures, as they can sometimes be seen in the freezers for other computer systems, the TURBO FREEZER represents the best possible solution that can be achieved with today's technology. And the invested effort was not in vain, as the result proves.

If you know the cumbersome primitive freezers of other computers you will simply be excited by the easy and instant usage of the TURBO FREEZER. For example freezing a game, saving it to the RAM extension and resuming the game three seconds later takes no more than three keystrokes. The same is true for loading and resuming the game from the RAM extension later on.

But the freezer can do much more. It is possible to perform any conversion between cassette and disk. Users of cassette tape drives who recently bought a disk drive can take their beloved software collection with them on disk. Users of disk drives can now use programs that are only available on cassette without tedious loading or can save money by buying the cheaper cassette versions.

And there is one more important point. The TURBO FREEZER is a freezer which has a DOS and debugger built in. They are always available and can be used without damaging the frozen program. Full disks or other fatal events are now no longer a problem anymore – even not when using application programs without DOS functions. The same is true for the evil bugs which can never be located without insight into the hardware registers and the unmodified system state.

2.1 Starting the freezer main menu

After pressing the "**Freeze**" button the program is frozen when the next interrupt occurs and the freezer takes over control of the Atari. From the freezer main menu all other functions can be reached. Resuming the frozen program is possible by pressing [SPACE]. The program will continue to run from exactly the same position where it was frozen. The freezer main menu offers the following options:

[SPACE] Resume frozen programm

[RESET] Perform cold start and deactivate freezer

[S] Save frozen program to disk or cassette

[F] Save frozen program to freezer RAM

[R] Save frozen program to RAM extension

[E] Load and start frozen program from disk or cassette

[C] Load and start frozen program from freezer RAM

[X] Load and start frozen program from RAM extension

[W] Swap frozen program with program in freezer RAM and start

[A] Swap frozen program with program in RAM extension and start

[Z] Clear RAM under the OS ROM

[D] Start debugger

[K] Start menu of the cartridge emulation

[Shift]+[K]
Start menu of the cartridge emulation without clearing the RAM (warm start)

[Shift]+[SPACE], [E], [C], [X], [W], [A]
Like above, but clear player/missile collision registers before resuming

[Control]+[E], [C], [X], [W], [A]
Like above, but frozen program is only loaded and not started

2.2 Freezing and resuming a program

In principle, any program can be frozen at any point even during a disk or cassette operation. Yet this is not recommended as this operation will remain incomplete. After resuming the program the operation system has the opportunity to retry the disk operation but this is not safe. Retrying an operation on the cassette is even impossible for the operation system since the cassette tape drive is simply too "stupid". Therefore it is better to freeze programs only when there is no operation with peripherals in progress.

In very rare cases nothing happens when the "**Freeze**" button is pressed and the program continues unaffectedly. This is the case when the program does not use interrupts at all. It's only possible if the program is very simple like for example converted Apple programs which do not really use the capabilities of the Atari. To also freeze such programs you

can install a [SYSTEM RESET] key in the Atari as described in section 1.5.2. The interrupt triggered by this key can by no means be suppressed, so there is no counter measure against the TURBO FREEZER anymore.

When a frozen program is resumed with [Shift]+[SPACE], the player/missile collision registers are cleared before control is handed over to the frozen program. Usually this function is not required, but in some games this prevents losing a life immediately after resuming. When loading frozen programs the same function is also available. Simply press [Shift]+[E] / [C] / [X] / [W] / [A] to apply it.

When a frozen program is loaded from disk, cassette, freezer RAM or RAM extension it is resumed by default. When loading a frozen program with [Control]+[E] / [C] / [X] / [W] / [A] it will not be resumed automatically. Instead you stay in the freezer and can for example change memory locations before resuming. This comes very handy if you search for the memory location that contains the number of lives, the amount of energy and so on. To this end you can freeze the program and save it immediately. Then you change the memory location and resume the program. If you used the wrong memory location, simply activate the freezer and load the frozen program with [Control]+[E]/[C]/[X] and try the next memory location. This way you have the same starting point and you don't have to worry about undoing the previous changes.

2.3 Saving a frozen program

The functions [S], [F], [R] can be used to save frozen programs to external mass storage like cassette or disk ([S] key), to the RAM extension ([R] key) or to the freezer RAM ([F] key). When using [S] a submenu will prompt if the frozen program shall be saved to cassette, as a boot disk or as a single file. Insert a cassette resp. disk before confirming your choice. When saving as a single file to a disk an additional menu will prompt for the file name.

Caution: When you save the frozen program as a boot disk, the disk that is present in "D1:" will be overwritten directly. All data possibly present on the disk will be lost in this case.

If you simply press [RETURN] without entering a file name, the file name CORE for "core dump" will be used. As a special feature wild cards can be used to overwrite an existing file. If you omit the D:, D1:, D2: etc. at the begin of the file name, D1: will be used automatically. The freezer supports up to 8 disk drives D1: to D8:.

In order to achieve the maximum possible speed the freezer menu is sometimes disabled during I/O operations. By accepting that the screen may "jump" when it's switched back on again, saving to the RAM extension only takes half of the time, what legitimates this dirty method.

2.3.1 Adapting the boot disk loader

When a frozen program is saved as boot disk, the freezer also writes a loader to the disk. This inherently will not work with every program because the loader itself requires about 2k of memory. If the frozen program requires already the complete memory of the Atari, it will not work together with the loader.

By default the loader uses the area between $C000 and $C6FF. This normally guarantees correct operation with all programs that only require 48k of memory. The start address of the loader can be adapted easily in case a program still runs into problems.

The first byte of the boot sector contains the start page of the loader (by default $C0). Adapting this byte is very easy with the TURBO FREEZER. Start the debugger, load the first sector, change the byte to the desired value and write the sector back to the disk. Please note that only values from $05 to $C9 and from $D8 to $F9 are allowed. If you use other values, you will obtain a `BOOT ERROR` when booting the disk. The following sequence of commands can be entered in the debugger to adapt the start page of the loader for example to $F0:

```
SR 1
C D700<F0
SW 1
```

2.3.2 Freezing a program that loads parts later

Freezing programs that load parts later is of course always possible. The only thing to remember is that the original disk must be inserted into the disk drive after resuming the program and before any further operation is performed in the running program. This of course wears out the disk over time. Therefore it is desirable to have a backup disk at hand so the original disk can be locked away in a safe location. But even with floppy speeders that include a copy function not all new programs can be copied by far.

The TURBO FREEZER does not remove the copy protection from the original disk, it only saves the current program state. In order to resume from this state later as often as you like, the original copy protected disk must be inserted into the disk drive again, before the program is resumed. If you use an additional disk to save the frozen game state, the original disk should be equipped with a write protection. This prevents accidentally overwriting the original disk.

If the game loads additional parts from multiple disks without copy protection (e. g. an adventure), it is advisable to copy these disks and use the copies instead to not wear the original disks out.

2.4 Swapping the frozen and a saved program

The swap function in the freezer main menu can be used to swap the current frozen program with the program that was saved to the freezer RAM ([W] key) or the RAM extension ([A] key) before. This way you can jump quickly between two different programs. You should be sure to really have a frozen program in the freezer RAM resp. in the RAM extension, otherwise the Atari will crash when resuming.

The modifier keys [Shift] for clearing the player/missile collision registers and [Control] for loading without starting automatically can be used with the swap function just like with the normal functions for resuming.

2.5 Clearing the RAM under the OS ROM

Because there are programs which require and also use 64k RAM, the freezer must also take the RAM under the OS ROM into consideration. After switching the computer on, this area is filled with useless random data. Because this data is ineligible to compression, a certain inefficiency and waste of space are the consequence when saving 48k programs to an external medium. In order to avoid this, the RAM from $C000...$FFFF can be cleared with the function [Z], in case the program only uses 48k RAM. The result is a reduction of the file size of about 25% to 50% and consequently shorter loading times.

The function can be used before booting the program or before saving the program. The first alternative is recommended if you are not completely sure that the program really does not store anything in this RAM area.

2.6 Starting debugger and DOS

The function [D] starts the built-in debugger with DOS functions. The debugger uses a command line and the complete screen, so controlling it via a menu would not be sufficient. Yet no RAM in the Atari is used or modified by this function. The states of the frozen hardware registers of course also remain intact. The complete description of the debugger and the DOS function can be found in chapter 3.

2.7 Starting the menu of the cartridge emulation

The function [K] starts the menu of the cartridge emulation. This function is only available if the "**CartEmu**" switch or the "**FlashWrite**" switch (or both) are on, i. e. are turned right. Otherwise the cartridge emulation is completely deactivated and consequently the menu of the cartridge cannot be started. The complete description of the cartridge emulation can be found in chapter 6. The freezer performs the following steps when the menu of the cartridge emulation is started:

- Leave the freezer main menu
- Activate the menu of the cartridge emulation
- Disable IRQs and NMIs
- Enable the OS ROM
- Set the return address to $E477 (cold start)
- Resume as usual (corresponds to pressing [SPACE])

With the function [Shift]+[K] a warm start is performed instead of a cold start (i. e. the return address set to $E474). This can be useful to keep the content of the internal RAM of the Atari when starting a module.

3 Debugger and DOS

Pressing [D] in the freezer main menu takes you to the built-in debugger with DOS functions.

Commands are entered in the command line at the bottom of the screen. Editing inside the command line is possible in the usual manner. The cursor cannot leave the command line. Using the arrow keys [CURSOR UP] and [CURSOR DOWN], the previous commands can be recalled. The freezer keeps the last 4 commands that were entered.

The debugger also supports the unfortunately little used [HELP] key. Pressing this key displays a short help with an overview of all commands. Using the keys [1], [2]... you can navigate to the individual help pages. Use [DEL] to display the previous page and [SPACE] or [HELP] to display the next page. Every other key ends the help display. The content of the command line remains unchanged when you enter the help. This means you can also open the help while entering a command if you forgot the exact syntax.

3.1 Disk Operating System (DOS)

Who didn't encounter the following situation: you have spent the past 3 your editing your program and now it's time to save it again. But instead of the expected success message you get a `File locked` or `Disk full` error message. Now it's hard to know what to do. Starting the DUP via `DOS` will make you lose your program because you didn't activate "MEM.SAV", as it takes forever to execute.

As the freezer is capable of stopping any program at any point, it suggests itself to also include a DOS with the most commonly used commands. Then it's no problem to handle the situation and continue the frozen program afterwards. The built-in DOS of the TURBO FREEZER contains all required functions to deal with the situation mentioned above. It supports single-, enhanced-, and double-density and is fully compatible to DOS 2.0 and DOS 2.5. The functions of the Turbo 1050 floppy speeder are fully supported, as well as the high SIO speed of Happy/Speedy compatible disk drives and XF-551 drives. Also any other tuned or standard disk drive can be used as well.

A disk command consists of a three character long command or a command followed by one or more spaces and a filename. Some commands also allow for specifying an option that may be added to the command, separated by a slash. The command must start immediately after the prompt. And besides the before mentioned space, no additional spaces must be entered. File names may contain the wildcard "*" replacing any character sequence, as well as "?" replacing any single character. Illegal commands will simply be ignored and will not cause an error message. Error messages are only be displayed if they were caused by the execution of a command.

3.1.1 Executing commands

If the prefix `D:` is not included in a file name, the default `D1:` will be used. RAM disks are not supported, as they always depend on the DOS and RAM disk driver that is used. Because of technical reasons PBI devices can unfortunately also not be used. Wildcards may also be used in the destination file names. In this case, the first matching file name is used. The only exception is the rename command `[REN]` which will be described in the next section.

The commands `DEL`, `LOC`, `UNL`, and `REN` for manipulating directory entries may operate on multiple files in a row. To prevent unwanted actions, an option has to be added to have the command be effective on multiple files with similar names. If no option is added, only the first matching file will be processed. The option `/Q` will ask for confirmation with `[Y]` for every matching file name. Pressing any other key than `[Y]` will skip the file. Stopping the command is possible using the `[BREAK]` key. And if you're sure, use the option `/A` to process all files with matching file names without confirmation. Errors that occur during execution will be displayed in readable text and will result in the return to the freezer main menu.

3.1.2 Command summary

The commands are listed here in table form only, as they are not new or unknown. Beginners may look up the commands in any DOS manual, like the one that comes with the Atari 1050 disk drive.

`DIR`
: List directory with all files

`DIR filename`
: List directory with certain files

`DEL filename`
: Delete file

`FMS`
: Format in single density

`FME`
: Format in enhanced density

`FMD`
: Format in double density, requires special or enhanced disk drive

`LOC filename`
: Lock file

`UNL filename`
: Unlock file

`REN filename,newname`
: Rename file

`LOA filename`
: Load object file

`LOA filename/N`
: Display object file load address(es) but do not load

`LOA filename,start`
: Load raw data file to address `start`, ignore COM header

`SAV filename,start,end`
: Save object file with the memory content from address `start` to (and including) `end`

`SAV filename/N,start,end`
: Save file without COM header (raw file) with the memory content from address `start` to (and including) `end`

3.1.3 Characteristics of specific commands

Some commands have specific characteristics compared to standard DOS commands. These characteristics are either improvements or are the logical result of working in a freezer environment.

While renaming files with `REN` command, the specification of the new file name may contain arbitrary wildcards. The wildcard characters are replaced by the characters from the original file name. This allows for working on groups of files efficiently, not being restricted to the primary or secondary name (i.e. the extension) of the file.

Object files can be loaded using `LOA`. The memory area to which the file is loaded is also displayed. In case of compound files, all memory areas are displayed. The complete 64k memory area can be used as target. This includes loading directly into the hardware registers, which are currently frozen. The loaded program will **not** be started, to prevent conflicts with the memory management logic of the freezer in case multiple segments are loaded.

Adding the option `/N` to the load command will display the memory area into which the program would be loaded but will not actually load the data into memory. This is useful in case you are only interested in the memory location to which an object file would be loaded, i. e.

```
LOA FONT.COM/N
```

The command `LOA` can also be used to load raw data (without COM header). In this case the address to which the data shall be loaded must be specified, i. e.

```
LOA FONT.DAT,8000
```

3.1.4 Error messages

When using the DOS and the load and save functions of the freezer for frozen programs, errors can occur. The corresponding error messages are displayed as plain English text.

`FILE NOT FOUND`
: The file could not be found

`FILE# MISMATCH`
: The (internal) file number does not match, the file structure is likely to be damaged. Rescue the other files if possible and format the disk

`BAD DISK I/O`
: The command cannot be executed due to a bus or disk error or because the write protection is active

`NO DRIVE` The disk drive does not respond

`DISK FULL`
: The disk is full

`FILE LOCKED`
: The file is locked

`DIRECTORY FULL`
: The directory is full (at most 64 files per disk)

3.2 Debugger

A debugger is used by machine language programmers to display, alter and improve object code and other memory contents directly in the computer memory. Doing this in the environment of a freezer brings certain advantages. Because of the way the freezer works, there are also some limitations that may be circumvented by using the right procedures.

3.2.1 Access to memory and hardware registers

Without a doubt, the biggest advantage is the possibility to work with the frozen system state. Doing this creates the impression of having a second computer, that's linked into the first, stopped Atari. It enables you to "look into" and change the stopped Atari and to resume any time. A debugger without freezer always has severe problems because of its own memory usage and because the I/O operations of the debugger itself change the system state. A programmers' term for this is "trashing", meaning turning the content of the system into "trash".

A standard debugger requires RAM for its own operation and will trash the data that was put there by the program that's being looked at. And even if the debugger is more sophisticated (and comes with its own RAM), it will still trash the hardware registers of ANTIC, POKEY and GTIA. There are no OS shadow registers for player missile graphics and sound registers and the existing OS shadow register will be deactivated by many programs anyway. As a result of the trashed hardware registers, it will be a lot of work, if even possible, to resume a program. And apart from this, debuggers without a freezer don't have the ability to display the content of the non-readable (write-only) hardware registers. So even if the register contents are not trashed by the debugger, the values in the registers remain unknown.

Using the built-in debugger of the TURBO FREEZER, all information about the system state is available. The contents of the hardware registers (i.e. what has been written there, not the status returned by a read operation) can be viewed in the I/O area and may be changed, without having to fear a system crash. The changes are applied only to the frozen program. You can work with the complete RAM area (even the RAM under the OS ROM), without trashing or fearing a crash.

In addition, there are various advantages resulting from running in the environment of the TURBO FREEZER. The load and save functions and the DOS functions allow for instantaneous testing and retrying, without lengthily reloading a program. And if the change works not as expected, it may be undone instantly without problems. Using this, even horrendous code monsters that overwrite parts of the DOS and that cannot be processed in a simple way, are not frightening anymore.

The software of the original TURBO FREEZER from 1987 had some disadvantages, among other reasons due to size constraints. Changes were always only applied to the frozen system state and changes to the hardware registers were only visible after resuming. This made it hard for example to work with the RAM extension.

The extensions of the software that even been implemented over time have meanwhile removed the majority of the original constraints. Using the direct I/O mode, the debugger can access extensions in the custom chip area and bank switching modules directly. And the `PB` command allows direct control of the access to the RAM extension, the OS ROM and the RAM under the OS ROM.

3.2.2 Command summary

The set of commands is kept minimal and is aligned with the Atari "Editor Assembler Cartridge" which makes it similar to many other debuggers. Hence a tabular summary of the command should suffice. The commands must not contain any spaces and must start immediately behind the prompt. Only the command line can be used for input. All changes are logged in the output area. All entries are in hexadecimal notation. By omitting a value, memory locations or registers may be left unchanged.

Example: `C100<0A,,4D` will change the content of the memory location $100 to $0A and of the memory location $102 to $4D. After this, the internal address counter will point to $103. The content of the memory location $101 remains unchanged.

For all commands, the address may be omitted and the debugger will use the internal address counter or an end address that will yield a reasonable output. The output can be stopped at any time by pressing `[S]` and resumed by pressing `[Q]`.

`Q` Go to freezer main menu

`D start` Display 8 bytes plus ATASCII characters starting at address `start`

`D start,` Display 128 bytes starting at address `start`

`D start,end`
Display memory content from address `start` to `end`

`I start` Display 8 bytes plus characters in internal Atari screen code starting at address `start`

`I start,` Display 128 bytes starting at address `start`

`I start,end`
Display memory content from address `start` to `end`

`L start` Disassemble starting at address `start`, display one screen page

`L start,end`
Disassemble from address `start` to `end`

`DL start` Display display list starting at address `start`

`DL start,end`
Display display list from address `start` to `end`

`C start<byte1,byte2...`
Change memory content starting at address `start`

`VEC` Display OS vectors

`HAT` Display handler table

`M` Display memory usage map

`PB` Display content of the memory management register `PORTB` (controls access to RAM extension, OS, BASIC)

`PB<value` Set content of the memory management register `PORTB` to `value`

`DIO` Display direct I/O mode

`DIO<value`
Activate/deactivate direct I/O mode, use with caution

`R` Display registers

`R<value1,value2...`
Change registers

`G start` Set return address for resuming to `start`

`/start,end/value...`
Search for `value...` in the memory area from address `start...end`

`BM start,end,target`
Move the memory area from `start...end` to `target...target+(end−start)`

`BS start,end,value`
Set memory area from `start...end` to `value`

`BC start,end,start2`
Compare memory area from `start...end` to the memory area starting at `start2`

`PR<value` Activate/deactivate printer output

`; TEXT` Print a text or comment

`SR number`
Read sector `number`

`SW number`
Write sector `number`

`SIO ...` Execute SIO command

`SIOR` Reset high speed SIO routine

`V` Display interrupt and content of the register `VCOUNT` at the time of the freeze

`V<value` Set the content of the register `VCOUNT` for the time when program will be resumed

`a COMMAND...`
AtariSIO remote control

`VER` Display freezer software version

Remark: With the exception of the spaces immediately following the command, all spaces in the above list are only there for readability and must not be entered with the command. The program counter (PC) cannot be changed using `R<` due to its special handling. Use the `G` command to change its value.

3.2.3 Input of values (hexadecimal, decimal, internal code)

Addresses and byte values are input in hexadecimal notation by default. Use the percent sign "`%`" as a prefix to enter an address or byte value as decimal number. The following command corresponds to the statement "POKE 710,10" in BASIC.

```
C%710<%10
```

Byte values can also be given in ATASCII or internal Atari screen code. For ATASCII, the single quote "`'`" must be used as prefix, for screen code the at sign "`@`" must be used.

```
C0600<'H,'a,'l,'l,'o
C9C40<@S,@c,@r,@e,@e,@n
```

3.2.4 Input of vectors

For working with vectors (i. e. display list or interrupt vectors) an additional variant for specifying addresses is available. If the prefix "`*`" is used for an address, the 2 byte vector at that address is evaluated and the content of the vector is used as effective address. This can be used for example to display the current display list very effectively.

```
DL*230
DL*%560
```

3.2.5 Access to extended memory, RAM, ROM

The debugger offers full control over the memory management functions of the Atari XL/XE when accessing the memory content. The memory management unit is controlled by the register `PORTB` ($D301) of the PIA. This way it is possible to read and change the content of the RAM extension ($4000...$7FFF) or the RAM under the OS ROM ($C000...$CFFF, $D800...$FFFF) without problems.

The `PORTB` control only refers to the access from within the debugger. Changes do not affect the frozen program. To change the state after resuming, the frozen content of `PORTB` must be changed using `C D301<byte`.

When the freezer is activated, the current value of `PORTB` is used as default for the debugger. That means you see the same state in the debugger as the frozen program would see. The `PORTB` control can be displayed and changed with the `PB` command:

`PB`	Display current `PORTB` value
`PB<value`	Set new `PORTB` value

If you would like to access the RAM under the OS ROM, enter `PB<FE` to set bit 0 of `PORTB` to 0. With `PB<E3` access to the first bank of the RAM extension in Atari 130 XE is activated. The content of `PORTB` at the time of the freeze can be displayed using `D D301` – provided it was not changed already using `C D301<value`.

The access to the frozen PIA registers ($D3xx) is subject to a special handling in the freezer. Normally, the values of the PIA registers are repeated every 4 bytes in the Atari. In the debugger, 8 bytes are displayed instead.

$D300, $D301
: are the normal registers `PORTA` and `PORTB`

$D302, $D303
: contain the data direction registers for `PORTA` and `PORTB`. They are normally accessible via `PORTA` resp. `PORTB` if bit 2 of `PACTL` / `PBCTL` is set to 0.

$D304, $D305
: contain the values of `PACTL` resp. `PBCTL` which are normally accessible via the address $D302 resp. $D303.

$D306, $D307
: are unused

3.2.6 Display display list

The `DL` command prints the ANTIC menmonics as described in the ANTIC data sheet, starting at the specified memory address.

`BLK x`	x blank lines
`CHR x`	Text mode x
`MAP x`	Bitmap mode x
`JMP adr`	Jump to given address
`JVB adr`	Wait for vertical blank, then jump to given address

Right to the mnemonic the following options are printed, if present:

`LMS adr`	Load memory scan counter (pointer to the current screen memory address)
`H`	Horizontal scrolling enabled
`V`	Vertical scrolling enabled
`I`	Trigger display list interrupt (DLI)

3.2.7 Search in memory

The search command can be used in various manners. Use `/start/value1,value2...` to search the specified byte sequence starting at the specified start address. At the first occurrence of the byte sequence, the search stops and prints the address. The internal address counter is set to the found address, so you can disassemble from that address immediately using the `L` command. The search can be resumed with the `/` command.

The byte sequence can consist of up to 8 bytes, which is more than sufficient in most cases. It's also possible to omit bytes from the sequence. These bytes are ignored by the search. `/1000/8D,,D4` will find the first address starting from $1000 that contains the command "STA $D4xx" to write a byte into the ANTIC.

The byte sequence can also contain a bit mask. To do this, the separator `&` followed by a bit mask must be appended to the byte, like for example `03&0F`. The content of the memory will combined with the bit mask using AND and will then be compared with the byte value from the byte sequence.

If an end address is specified in addition to the start address, all addresses where the byte sequence is found are printed. That means the search does not stop at the first occurrence when entering for example `/start,end/value1,value2...`. Many times, the complete memory is to be examined. Therefore there is the very short variant `//value1,value2...` which is identical to `/0000,FFFF/value1,value2...`.

If the byte sequence is long or the memory area is large, the search may take several seconds. Using the `[BREAK]` key, the search can be interrupted at any time. It can then be resumed with the `/` command.

3.2.8 Execute SIO commands

The commands `SW`, `SR`, `SIO` as well as the built-in DOS use an internal sector buffer. The memory manager mirrors it to the address $D700 in the frozen address space, where it can be edited by the debugger. Physically it is not present at that address. To read a sector from a different drive than `D1:` or to write it there, just prepend the sector number with `Dx:`.

```
SR 100
SR D2:200
SW D3:300
```

With the `SIO` command, arbitrary commands can be sent to the SIO, just like using the SIO vector $E459. The values have the same meaning as the memory locations $0300...$030B in the Atari. The maximum allowed value for `length` is $0100.

```
SIO device, unit, command, direction, timeout, length, daux
```

The `SIO` command retains the parameters of the previous execution, so they can be omitted in the next `SIO` command. If all parameters are omitted, the last `SIO` is repeated. At the beginning, the parameters are set to "Get Status" for `D1:`, so nothing wrong can happen in case `SIO` is entered without parameters.

In any case the `SIO` should always be used with caution and the parameters should be verfied thoroughly, because otherwise formatting the wrong disk may happen quickly.

3.2.9 Display interrupt and VCOUNT

The freezer can freeze the running program when an interrupt occurs. To this end the freezer reroutes the interrupt vector and thereby forces the CPU to enter the freezer. The

`V` command displays the type of interrupt (IRQ or NMI) that caused activation of the freezer, together with the original value of the corresponding interrupt vector. Additionally the value of `VCOUNT` at the moment of the freezer activation as well as the calculated value of `VCOUNT` at the moment the interrupt occurred are displayed. This command was mainly introduced to test the freezer software and may be of little interest for most users. But here's a short description of the significance and origin of the values.

When the freezer is activated, it maps the ROM with the freezer software into the address space of the CPU and "reroutes" the high byte of the interrupt vector to the ROM's address, so the freezer software is started. Because the low byte is not changed, there's a full page of "NOP" instructions at the start of the freezer software. Therefore it may be the case that the Atari has to execute through several "NOP" instructions to get to the part that will read the value of `VCOUNT`.

Using the low-byte of the interrupt vector, the software tries to calculate the amount of time it took to get through the "NOP" instructions and adjusts the `VCOUNT` value accordingly. This way the program can be resumed at exactly the same position where it was interrupted. This calculation is not 100% correct and may vary according to the graphics mode. In the very rare case that the value is not correct and causes problems, the command `V<value` can be used to set the `VCOUNT` value manually.

3.2.10 Activate direct I/O mode

Changes to hardware registers only become effective after resuming, because in the TURBO FREEZER only the frozen system state is manipulated, as opposed to normal debuggers which manipulate the real system sate. For experts the TURBO FREEZER offers the possibility to activate the "direct I/O mode" and manipulate the real system state also in the debugger.

`DIO`	Display current mode: `0` = deactivated (default), `1` = activated
`DIO<0`	Deactivate direct I/O mode
`DIO<1`	Activate direct I/O mode

Caution: When direct I/O mode is active, the freezer is "bypassed" and it becomes likely to make mistakes. The freezer will not notice any changes to the hardware registers in direct I/O mode. Therefore the same changes should also be performed again with direct I/O mode switched off. Otherwise the freezer will overwrite the changes when resuming.

3.2.11 Display version number

The command `VER` prints the version number and the date of the freezer software in the format `3.10 2012-11-09`.

3.2.12 Display OS vectors

The `VEC` command displays the most important OS vectors of page 0 and 2 as an overview. The output contains the symbolic name of each vector and its value.

```
DOSINI 000C: 0000   CASINI 0002: FFFF
DOSVEC 000A: F223
VPRCED 0202: C0CD   VINTER 0204: C0CD
...
```

3.2.13 Display handler address table

The `HAT` command lists the entries of the handler table ($031A...$033A). It displays the address of the entry, then the 3 bytes of the entry followed by the device letter and the address of the handler table (resp. "`-- 0000`" if the entry is empty).

```
031A  50 30 E4  P: E430
031D  43 40 E4  C: E440
0320  45 00 E4  E: E400
```

3.2.14 Display memory usage map

The command `M` displays a map of the memory which indicates used and unused memory pages. If a page contains only zeros, a period "`.`" is displayed. If it contains at least one non-zero byte, a star "`*`" is displayed. This is useful for example when searching for an empty memory area for the boot loader (see section 2.3.1).

3.2.15 Activate printer output

Printer output is controlled with the command `PR<value`. Values from `1` to `8` activate the output to printer `P1:` (default printer) to `P8:`. The value `0` deactivates printer output. Via `PR`, the current state of the printer output is displayed on the screen. If printer output if active, everything that is displayed on the debugger screen is also printed on the specified printer in parallel. This is a handy way of logging debugger sessions.

Especially for longer debugger sessions it can be very useful to insert comments into the logs. This can be achieved easily with the "`;`" command of the debugger. Everything that follows the semicolon is printed in the debugger window (and hence, if printer output is activated, also on the printer). This way you no longer need to grab pen and paper frequently and the logs of the debugging session will be understandable even months later.

3.2.16 Control AtariSIO remotely

Using the `a` command, remote control commands can be sent to AtariSIO. This enables a complete remote control of AtariSIO without loading and additional program. More information can be found in the AtariSIO manual.

4 512k RAM extension

The built-in RAM extension of the TURBO FREEZER is compatible with the wide spread Rambo/ATARI-Magazin extensions. Bit 4 of `PORTB` activates access to the extension (0 = extended RAM at $4000...$7FFF enabled, 1 = internal Atari RAM enabled). The bits 2,3,5,6 and 7 select one of the 32 16k banks. A separate ANTIC access, as it is present in the Atari 130 XE and some other extensions, is not supported. On the one hand, the required HALT signal is not available on the PBI bus. On the other hand, there are only very few demos which require this mode. And meanwhile most of them have been patched to work with RAM extensions that do not support separate ANTIC access.

The following restriction may arise with programs which require a RAM extension with separate ANTIC access. If a RAM extension which supports separate ANTIC access is present in the Atari, activating the built-in 512k RAM extension of the freezer may cause malfunction. If ANTIC access to the RAM extension is enabled (via bit 5 of `PORTB`), but CPU (or combined CPU/ANTIC) access is disabled (via bit 4 of `PORTB`), the ANTIC will access the internal RAM extension instead of the RAM extension of the TURBO FREEZER. The best solution is to prevent this situation by disabling either of the RAM extensions.

5 Oldrunner – Atari OS Rev. B

The Atari 400/800 is the only 8-bit home computer class that contains a well-designed and structured operating system (OS). This allows for changes, extensions and improvements without the need to adapt current programs. That's why Atari could use a more sophisticated OS in the later Atari XL/XE series. Annoyingly, there are programs (from 1980-1983) that don't run on the XL series. This is the fault of the programmers who did not stick to the official programming guidelines.

To be able to use these incompatible programs on the Atari XL/XE, so called "Translator" disks were published. They disable the OS ROM and copy the old OS version "Rev. B" into the RAM which is located under the OS ROM. Although this works well in most cases, it's time consuming and unpleasant to always have to load this alternative OS. Only a hardware solution where the old OS is stored in a kind of ROM that is always available and unmodifiable will work with all incompatible programs. Unfortunately these so called "Oldrunners" normally require modifications within the Atari.

With the TURBO FREEZER it is now possible to implement an Oldrunner without modifications within the Atari. The memory management logic makes this possible. The Oldrunner can be activated and deactivated using the switch labeled "**OldOS**". Move the switch to the right position `ON` to activate the Oldrunner. Move the switch only while the Atari is off. Otherwise the content of the RAM will not match the required content of the newly selected OS version and the system will crash.

For various reasons, it's best to only activate the Oldrunner if it is the only way to get a program to work. While the Oldrunner is active, there is no built-in BASIC, no RAM extension and no warm start. Pressing the `[RESET]` key while the Oldrunner is active always (!) triggers a cold start and the content of the RAM will be lost. The key which corresponds to the `[RESET]` key of the Atari XL/XE was called `[SYSTEM RESET]` in the Atari 400/800 and triggered a non-maskable interrupt (NMI). It's not very difficult to retrofit this key (see section 1.5.2) but it involves opening the Atari and is therefore rather an expert task. Besides that, this key is not really needed anyway because most of the incompatible programs are games which have mapped the `[SYSTEM RESET]` to a cold start or a system crash. This is the ridiculous attempt to annoy the "crackers", but misses the point and only annoys the user instead.

6 Cartridge emulation

The cartridge emulation is another very powerful feature of the TURBO FREEZER. It is capable of emulating standard 8K and 16K modules as well as bank switching modules according to the AtariMax/MegaMax and OSS standards. Additionally the new SpartaDOS X (Ultimate1MB bank switching, up to 512k) can be emulated. Also combined usage of SpartaDOS X together with an OSS module is allowed.

The cartridge emulation can use all unused memory of the flash ROM (960k) and the freezer RAM (384K). This means the TURBO FREEZER gives you instant access to up to 168 different modules. The cartridge emulation is not only interesting for those who use many different cartridges and who became tired of changing the cartridges all the time (which heavily strains the cartridge port), but also for those who develop cartridge based software. Because the module data can not only be stored in the flash ROM but also in the freezer RAM, changes to the data can be performed and tested within seconds. Since the freezer RAM is battery backed its content is not lost after switching the Atari off. This means the freezer RAM can be used for storing permanent data, just like the flash ROM.

6.1 Basics

Leveraging the full potential of the cartridge emulation requires knowledge about the following details of the internal operation.

6.1.1 Bank numbers

The flash ROM is internally divided into "banks" of 8K. With 1MB flash ROM, there are 128 8k banks with the bank numbers 0...127 available. The topmost 64k of the flash ROM are used by the freezer software and cannot be used for cartridge emulation. This leaves the banks with the numbers 0...119 for cartridge emulation.

In the 512k freezer RAM the topmost 128k are reserved for the freezer software, leaving 384k in the banks 0...47 for the user. Snapshots are stored in the banks 48...55 of the freezer RAM. If you don't use snapshots, you can use these banks also for cartridge emulation.

16k modules use 2 consecutive 8k banks. Their data must be aligned to a 16k boundary, i.e. an even bank number. If a 16k module is for example stored in the banks 2 and 3, then bank 2 will be visible at $8000...$9FFF and bank 3 will be visible at $A000...$BFFF.

512k modules, SpartaDOS X and modules with TURBO FREEZER 2005 bank switching (`8k old`) must be aligned to a 512k boundary. That means they can only start at bank 0 or 64 in the flash ROM or at bank 0 in the freezer RAM.

The SpartaDOS X emulation uses the same bank switching scheme as the "Ultimate 1MB" extension. The corresponding image has a maximum size of 512k. Currently the SpartaDOS X 4.45 image for the Ultimate 1MB extension has 256k and can therefore be used without problems at bank 0 or 64 in the flash ROM or at bank 0 in the freezer RAM.

6.1.2 Module types

The module type defines at which address in the Atari the data from the cartridge emulation shall be visible. Also additional bank switching registers (e.g. for OSS modules) are activated the area $D500...$D5FF, depending on the module type.

The following module types are supported:

`8k` 8k standard module, $A000...$BFFF.
The software can access the whole memory, i.e. the flash ROM and the freezer RAM, via the bank switching register of the cartridge emulation.

`8k+RAM` 8k module, $A000...$BFFF
with optional 8k RAM bank at $9000...$BFFF. Both areas can be selected independently via separate bank registers.

`8k old` 512k bankswitching module, $A000...$BFFF.
TURBO FREEZER 2005 compatible bank switching.
Bank 0 or 64 must be used as start bank.

`8k AtariMax`
1MB (8MBit) bank switching module, $A000...$BFFF.
Bank switching according to the AtariMax/MegaMax standard.

Caution! The cartridge emulation offers 64k less than the AtariMax 8Mbit module. Therefore modules which require the complete 1MB of memory will not work, like for example Space Harrier. When creating own modules with the MaxFlash software, the maximum size of 960k must not be exceeded. In addition the bank 0 must be used as start bank.

`16k` 16k standard module, $8000...$BFFF.
Bank switching is possible like in the 8k mode.

`OSS` 16k OSS bank switching module, $A000...$BFFF.
For example MAC/65 or ACTION!.

6.2 Activation of the cartridge emulation

To activate the cartridge emulation, move the "**CartEmu**" switch to the right position (`ON`) while the Atari is switched off. When switching the Atari on, a small menu is displayed where the cartridge emulation can be configured.

Additionally you can enter the cartridge emulation menu from the TURBO FREEZER menu via the key `[K]` or `[Shift]+[K]`. This works only if the cartridge emulation is currently active, i.e. at least one of the switches "**CartEmu**" or

"**FlashWrite**" is in the right position (ON). The cartridge emulation is completely disabled if both switches are in the left position (OFF).

6.3 Cartridge emulation menu

The current configuration is displayed in the middle of the screen. The online help with the available options is displayed at the bottom of the screen.

MODE — Selects the emulated module type.
OFF deactivates the cartridge emulation.
PicoDos starts the integrated MyPicoDos instead of a module.

SRC — Selects the flash ROM or freezer RAM as source for the module data.

BANK — Selects the flash ROM or freezer RAM start bank.

SDX — Selects the SpartaDOS X emulation. You can choose between OFF, ROM Bank 0, ROM Bank 64 and RAM Bank 0.

BOOT — Selects if the module shall be started via a cold start (COLD) or a warm start (WARM). This option should normally be set to COLD, because otherwise the operating system variables and the module are not correctly initialized. The option WARM is intended for developers and only makes sense if the cartridge emulation was activated from the TURBO FREEZER menu via [Shift]+[K]. This way the cartridge emulation can be reconfigured while keeping the internal RAM of the ATARI unchanged. This can be useful during development.

Some frequently used configurations can be selected with a single key press.

[D] — Selects the default configuration: Mode 8k, Bank 0, Source ROM, SDX OFF and Boot Cold.

[P] — Selects the Mode PicoDos.

[O] — Deactivates the cartridge emulation: Mode OFF, SDX OFF

Pressing the [RETURN] key activates the currently selected configuration. Pressing [ESC] deactivates the cartridge emulation and restarts the Atari.

6.4 Command summary

The debugger supports the following commands, which all begin with K, to control the cartridge emulation. The commands have the same effect as manually chaning the configuration registers starting at $D500, but some people are said to prefer mnemonics over manual entry of hexadecimal values.

`K` Display cartridge emulation configuration

`K<0` Deactivate cartridge emulation completely:
Sets module type and SpartaDOS X emulation to `OFF`, all banks to `0` and source to `ROM`

`KM<type` Sets module type of the cartridge emulation. Possible values:
- `0` off
- `8` 8k module
- `8R` 8k module with optional 8k RAM bank at $8000
- `8O` 8k module with TURBO FREEZER 2005 bankswitching
- `A` 8k AtariMax/MegaMax module
- `16` 16k module
- `O` 16k OSS module

`KB<bank` Select 8k main bank, ($00...$7F).

`KBE<mode` Switch 8k main bank on/off:
- `0` off
- `1` on

`KR<bank` Select optional 8k RAM bank ($00...$3D)

`KRE<mode` Switch optional 8k RAM bank on/off:
- `0` off
- `1` on

`KX<mode` Select SpartaDOS X emulation mode:
- `0` off
- `1` on and in flash ROM at bank 0
- `2` on and in flash ROM at bank 64
- `3` on and in freezer RAM at bank 0

`KXB<bank` Select SpartaDOS X emulation bank ($00...$3F). Only possible, if emulation was activated via `KX`

`KXM<mode` Select SpartaDOS X mode. Only possible, if emulation was activated via `KX`
- `0` SpartaDOS X off, additional module off
- `1` SpartaDOS X on, additional module off
- `C` SpartaDOS X off, additional module on

`KO<bank` Select OSS bank. Only possible, if emulation was activated via `KM<O`
- `0` OSS module off
- `1...3` Select OSS bank `1...3`

`KW<mode` Switch write access to cartridge emulation on/off:
- `0` off
- `1` on

`KS<source`
: Select source for the cartridge emulation:
  `0` flash ROM
  `1` freezer RAM

`KN<mode` Switch menu for cartridge emulation on/off:
  `0` cartridge emulation normal, menu deactivated
  `1` cartridge emulation off, menu activated

6.5 Further information

6.5.1 Using "real" cartridges

When using a "real" module in the cartridge port of the Atari 600 XL / 800 XL or in the cartridge port of the Atari XE adapter board of the TURBO FREEZER, the cartridge emulation should be deactivated completely. To this end the "**CartEmu**" as well as the "**FlashWrite**" switch must be moved to the left position (`OFF`). Otherwise the system may not work correctly, particularly if the cartridge uses bank switching.

6.5.2 TRIG3 – $D013

Compared to the cartridge emulation of the TURBO FREEZER 2005, a small but important detail in the cartridge emulation of the TURBO FREEZER 2011 has been improved. If the cartridge emulation is active, the presence of a module is correctly reported when reading the `TRIG3` ($D013) register. This means the new cartridge emulation behaves 100% like a "real" module and patching of module images, which was necessary for the TURBO FREEZER 2005, is no longer required.

7 Programming the flash ROM

7.1 Basics

The software of the TURBO FREEZERs is not, as this is normally the case, stored in an EPROM but in a flash ROM. The big advantage of the flash ROM is that it can be re-programmed directly via the Atari. This means software updates can be installed at any time and the unused parts of the flash ROM can be filled with module data without the need for a special programming device. The flash ROM in the TURBO FREEZER can be re-programmed up to 1.000.000 times, which should be enough for most experiments.

The flash ROM and the freezer RAM of the TURBO FREEZER are normally protected against overwriting. The "**FlashWrite**" must be moved to the right position (`ON`) to enable write access to both memories. If the switch is in the left position (`OFF`), the flash ROM and the freezer RAM are write protected. It is very unlikely that any software overwrites the flash ROM accidentally. To program the flash ROM a special write sequence must be sent to the flash ROM, otherwise write access is simply ignored. Therefore is not really dangerous to have the "**FlashWrite**" switch in the right position (`ON`) permanently.

When using "real" bank switching cartridges in the cartridge port both the "**CartEmu**" as well as the "**FlashWrite**" switch should be turned `OFF`. This prevents a conflict between the cartridge emulation and the bank switching cartridges.

Caution: It is important that the "**OldOS**" switch is turned `OFF` during flashing in order to ensure that the flash ROM is exclusively accessible by the flashing program. If the Oldrunner is active at the same time conflicts can occur – e. g. when an interrupt is raised – and the Atari can crash.

7.2 Banks and blocks in the flash ROM

It is helpful to know some details about the inner structure of the flash ROM in order to use it best. The used flash ROM is partitioned into blocks of 64k each (i. e. eight 8k banks). These blocks can be re-programed completely independent of each other. This means that e. g. changes to the data of the cartridge emulation to not impact the freezer software and vice versa.

Unfortunately the partitioning into 64k blocks also has a small disadvantage. It is not possible to re-program only a part of a 64k block. Still free, not yet programmed parts can be programmed later on. You can for example first program an 8k module into bank 0 and program later another 8k module into bank 1 of the same block. But is it recommended to merge several e. g. 8k modules into a large file and program them then in one go. The

restriction to 64k blocks does of course not apply to the freezer RAM. There every 8k bank can be programmed separately.

If the start bank is not at a 64k block boundary, then the flash program asks if the complete 64k block shall first be erased or not. If the 64k block is not erased, unused (e. g. previously erased) parts of a 64k block can be filled with data. If you accidentally try to overwrite already used parts, an error message appears and you have to re-program the complete 64k block.

7.3 Starting the flash program

To program the flash ROM resp. the freezer RAM the "**FlashWrite**" switch must be turned right (`ON`). The program `FLASH.COM` must be loaded from the system disk. If the switch is turned `OFF`, then the program will not find the flash ROM and will output an error message. In this case you can turn the switch right and confirm the question `restart program?` with `[Y]`. The flash program outputs the following information when it starts:

- Type of the flash ROM
- Version number and date of freezer software in the flash ROM

In case no freezer software is present in the flash ROM, the output reads `n/a` instead of the version number. The version number can also be displayed in the debugger via the `VER` command.

7.4 Using the flash program

The flash program offers the following operations:

`Program CartEmu flash`
: Program the flash ROM area of the cartridge emulation (banks 0...119). The area for the freezer software (banks 120...127) remains protected in this case and cannot be overwritten accidentally. If you only work with the cartridge emulation, then you should always only use this operation.

`Program CartEmu+Freezer flash`
: Program the complete flash ROM (banks 0...127). With this you can update also the freezer software (banks 120...127).

`Write flash to file`
: Write the flash ROM content to a file.

`Program RAM`
: Program the freezer RAM (banks 0...47).

`Write RAM to file`
: Write the freezer RAM content to a file.

`Erase flash`
: Erase the complete flash ROM.

  Caution: This will erase the complete flash ROM, including the freezer software. In order to then use the TURBO FREEZER in a normal fashion you have to first re-program the freezer software again starting at bank 120. Otherwise the Atari will crash if the "**Freeze**" button is pressed or if the cartridge emulation or the Oldrunner is activated.

`Start CartEmu`
: Jump to the menu of the cartridge emulation. With this you can test the module you have just programmed.

7.5 Programming the flash ROM and freezer RAM

The following steps are performed when programing the flash ROM resp. the freezer RAM. At first the flash program asks the user for the first bank number to be programmed. Valid values are 0...127 for the flash ROM and 0...47 for the freezer RAM. Then the flash program asks for the number of 8k banks to be programmed. If you confirm the default value `0`, the complete file will be programmed automatically.

Now the name of the file which contains the data to be programmed must be entered. The file must only contain the data to be programmed, i. e. it must not contain a COM header or something similar and the file size must be a multiple of 8k. The single 8k banks of the flash ROM (or the freezer RAM) are now programmed sequentially. When programing the flash ROM, every 64k block will be erased when a 64k block boundary is reached. Programming will also stop if the end of the flash ROM or freezer RAM area is reached before the end of the file is reached.

During keyboard input and during programming, the current operation can be aborted with the `[BREAK]` key at any time.

7.6 Updating the freezer software

If the freezer software in the flash ROM was overwritten or erased accidentally, or if the software shall be replaced by a newer version, then the file `FREEZER.ROM` from the system disk must be programmed to the banks 120...127 of the flash ROM. To this end the following steps are required:

1. Turn the "**FlashWrite**" switch right (`ON`).
2. Load the flash program `FLASH.COM` from the system disk.
3. Choose the 2nd menu item (`Program CartEmu+Freezer flash`).
4. Enter start bank `120`.
5. Confirm the default value `0` for the number of banks with `[RETURN]`.
6. Enter `FREEZER.ROM` as file name.

8 For experts: Technical details

8.1 Hardware

The heart of the TURBO FREEZER is the Xilinx XC95144XL CPLD. It contains the complete logic of the freezer and controls the freezer RAM as well as the cartridge emulation. There are 1MB flash ROM and 1MB battery backed freezer RAM present which are shared between the functional blocks, i.e. freezer, cartridge emulation and 512k RAM extension.

When the freezer RAM or the flash ROM of the freezer have to be mirrored into the Atari, the internal RAM and ROM of the Atari must be deactivated. This is achieved using a small trick. The freezer pulls the line of the ANTIC refresh pin, which is intended to be an output, to "low". This makes the MMU in the Atari believe that ANTIC is performing a memory refresh, hence it deactivates all built-in memory. According to Bernhard Engl, who works as a full-time chip designer nowadays, it is not dangerous to pull a "high" level "low" from outside in the NMOS-Depletion-Load technology, because that is how NMOS works anyway. If the ANTIC had been manufactured in CMOS technology, this approach would have been dangerous and could have damaged the ANTIC – and there never had been a TURBO FREEZER.

8.2 Memory partitions

8.2.1 Freezer RAM

512k of the freezer RAM are directly assigned to the 512k RAM extension. The remaining 512k RAM are shared between the freezer and the cartridge emulation. The freezer can access the upper 128k (banks 48...63) of this area. The topmost 16k (banks 62 and 63) are exclusively reserved for the freezer (i. e. for the hardware register shadowing). The bank switching in the freezer addresses the freezer RAM not in 8k banks, like the cartridge emulation, but in 4k banks. The cartridge emulation can access the complete RAM except for the last 16k, making available a maximum of 496k (banks 0...61).

8.2.2 Flash ROM

The cartridge emulation can access the complete 1MB flash ROM, the freezer can access the upper 512k thereof. The topmost 64k are reserved for the freezer software, the Oldrunner OS and the cartridge emulation menu. The topmost 64k (banks 120...127) are currently used as follows.

- Bank 120: Free
- Bank 121: Free
- Bank 122: Free
- Bank 123: Freezer software bank 3
- Bank 124: Freezer software bank 1 (start bank)
- Bank 125: Freezer software bank 2
- Bank 126: Cartridge emulation menu
- Bank 127: Oldrunner OS

The banks 127 (Oldrunner OS), 126 (cartridge emulation) and 124 (freezer software) are "hardwired" in the CPLD logic. The usage of the remaining banks 120...123 and 125 may change in later versions.

8.3 Freezer

As long as the "**Freeze**" button is not pressed, the freezer is completely invisible for the Atari. That means, no software can detect if there is a freezer attached to the Atari or not. Only when the button is pressed, the freezer becomes visible and active.

8.3.1 Shadowing of hardware registers

The freezer uses the following "trick" to read the content of the non-readable hardware registers of ANTIC, GTIA and POKEY. Whenever there is a write access to the area $D000...$D7FF, the freezer RAM is also activated. This way the freezer software can read the data of the custom chips from there without problems. The write access to the RAM is only performed if the "**Freeze**" button has not been pressed yet. Otherwise the freezer software itself would overwrite the stored data.

The addresses of custom chips in the Atari are not completely decoded. The ANTIC can for example be accessed via the addresses $D40x, $D41x, ... When $22 is written to $D400 and $00 is written to $D410 afterwards, both values will end up in the `DMACTL` register of the ANTIC. `DMACTL` will have the value $00 after the 2nd write access.

Some programs use these tricks. To enable the freezer to restore the exact state upon resuming, the freezer logic has to take this in account. Therefore the non-decoded address lines are pulled "low" when shadowing the hardware register. For ANTIC ($D4xx) and POKEY ($D2xx) these are A4–A7, for GTIA ($D0xx) these are A5–A7.

If the Atari contains stereo POKEY extension, then the first POKEY is accessible via $D20x, $D22x, ..., the second POKEY via $D21x, $D23x, ... Turning the "**Stereo**" switch of the freezer right (`ON`) changes the shadowing logic. During access to $D2xx the address lines A5–A7 are pulled "low" then and the data of both POKEYs are stored correctly.

8.3.2 Mirroring the freezer memory in

When the freezer is activated, the freezer RAM resp. the flash ROM is mirrored to the area $0000...$3FFF. Bank switching is used to choose between 32 RAM banks of 4k each (a total of 128k) and 64 ROM banks of 8k each (a total of 512k). The memory map in the Atari looks as follows.

- $0000...$0FFF: Fixed 4k RAM bank (bank 31)
- $1000...$1FFF: Switchable 4k RAM bank (bank 0...31)
- $2000...$3FFF: Switchable 8k ROM bank (bank 64...127)

The 64 ROM banks of 8k each are located at the end of the flash ROM and correspond to the banks 64...127 of the cartridge emulation. The 32 RAM banks of 4k each are located at the end of the freezer RAMs and correspond to the banks 48...63 of the cartridge emulation. The last 4k bank (bank 31) is used for the shadowing of the hardware registers and is mirrored to the address $0000 in the Atari if the freezer is activated.

8.3.3 Activating the freezer

The freezer logic has 5 internal states.

- Freezer deactivated
- Freezer half activated
- Freezer activation triggered
- Freezer activated
- Freezer temporarily deactivated

After switching the Atari on, the freezer is in the state “deactivated”. In this state and in the state “half activated” all write access to the custom chip area is stored in the shadow RAM.

When the “**Freeze**” button is pressed, nothing happens at first. Only when the first access to the area $FFF8...$FFFF (which contains also the IRQ and NMI vectors) is performed and the “**Freeze**” button is still pressed, the freezer reacts. It saves the state of the A2 line (which is required later on to distinguish between IRQ and NMI) and enters the state “half activated”. In the next clock cycle there are now these two possibilities:

- If the “**Freeze**” button is no longer pressed or an access to another memory area than $FFF8...$FFFF is performed, the freezer enters the “deactivated” state again.
- If the “**Freeze**” button is still pressed and there is an access to the memory area $FFF8...$FFFF again, the freezer performs the transition to the state “activation triggered” as described below.

The freezer deactivates the memory in the Atari and puts the value $21 on the data bus. This changes the interrupt vectors to the address $21xx, as the CPU is currently trying to load the high byte of the interrupt vector. The low byte stems from the previous access to the $FFF8...$FFFF area and therefore has the original value.

After this, the freezer enters the state “activation triggered”. Just like in the state “activated” the TURBO FREEZER now mirrors its own memory in. The ROM bank 124 with the start code of the freezer software is mirrored to the memory area $2000...$3FFF. Starting at $2000 a complete page full of “RTI” instructions is present. After this, i. e. starting at $2100 a so-called “NOP slide”, i. e. a page full of “NOP” instructions for the “rerouted” interrupt routine is present. The actual freezer software starting with the initialization routine follows after that.

During the activation phase, the occurrence of IRQs is no problem, because they are masked by the CPU during the execution of an interrupt routine anyway. If an NMI occurs during the activation phase, the freezer reroutes its interrupt vectors to the address $20xx. There only “RTI” are present and the NMI will be ended again immediately. This means the freezer “suppresses” all further interrupts during the activation phase.

After running through the NOP slide and disabling the interrupts the software puts the freezer into the state “activated” via a write access to the address $D720. If the now activated freezer software wants to access the Atari memory in the area $0000...$3FFF it first has to disable the freezer RAM resp. the flash ROM temporarily. This is done via a write access to the address $D700, which puts the freezer into the state “temporarily deactivated”. An additional write access to the address $D700 puts the freezer back into the state “activated” and the freezer RAM resp. the flash ROM are mirrored in again.

When the freezer software is exiting and the interrupted program is resumed, the freezer has of course to be put into the state “deactivated” again. This is done via a read access to the address $D700.

8.3.4 Configuration registers

When the freezer is activated, a number of configuration registers is mirrored into the area $D700...$D7FF. When these registers are accessed, the internal memory in the Atari is deactivated to prevent problems with potentially installed internal extensions.

$D700...$D70F: Freezer State Control (w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Write access to this register toggles the freezer between the states “activated” and “temporarily deactivated”.

$D700...$D70F: Freezer Disable (r)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read access to this register put the freezer into the state “deactivated”.

$D710...$D71F: Freezer RAM A4/A5 (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read access to this register changes the handling of the address lines A4 and A5 when accessing the freezer RAM ($0000...$1FFF). A4 is set to the state A2 had when the freezer was activated. A5 is set to the current state of the "**Stereo**" switch. This way the freezer software can detect if the freezer was activated via an NMI or via an IRQ and if the stereo POKEY mode is active or not. A write access to this register resets the logic to the normal handling again.

$D720...$D72F: Freezer Startup Complete (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this register puts the freezer from the state "activation triggered" into the state "activated".

$D740...$D77F: Flash ROM Bank Select (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to these registers selects the 8k flash ROM bank which is mirrored to $2000...$3FFF. An access to $D740 selects bank 64, $D741 selects bank 65, ...$D77F selects bank 127.

$D780...$D79F: Freezer RAM Bank Select (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to these registers selects the 4k freezer RAM bank which is mirrored to $1000...$2FFF. An access to $D780 selects bank 0, $D781 selects bank 1, ...$D79F selects bank 31.

8.3.5 Freezer software and OS ROMs

The freezer software uses its own code for the most functions (e. g. the keyboard input) instead of using the routines of the operating system. This makes the software mostly independent of the used operating system and also independent of the state of the Atari when freezing is triggered. Yet the used operation system must provide some functions to ensure correct operation of the freezer software and these functions must behave like in the original operating system.

- Standard character set at $E000...$E3FF
- NMI and IRQ handling
- SIO function ($E459), used by the `SIO` command in the debugger and when save to or loading from cassette

All alternative OS versions which have been available so far, like for example "QMEG OS" or "Romdisk OS", fulfill these requirements. Problems came up recently with the new

"Ultimate 1MB BIOS". It neither contains a standard interrupt handling nor a character set at $E000. Therefore the screen is completely garbled when the freezer is activated from within the "Ultimate 1MB BIOS".

8.4 512k RAM extension

A separate 512k RAM chip is included for the 512k RAM extension. This way the freezer, the cartridge emulation with modules in the freezer RAM, and the RAM extension can be used in parallel to the full extent.

Because the PIA signals are not available on the PBI, the CPLD must mimic the behavior of the PIA registers. The PIA in the Atari keeps working "in parallel", so BASIC, OS ROM, and Self Test are controlled as usual. Read access to the PIA is ignored by the CPLD, i. e. in this case the Atari reads the real PIA registers as usual. The logic only reacts on write access to the addresses $D301 (`PORTB`) and $D303 (`PBCTL`). The CPLD also saves the value of bit 2 of `PBCTL`, which controls if the access is performed on the data register `PORTB` (bit 2 = 1) or on the data direction register (bit 2 = 0).

The CPLD stores the data of a write access to $D301 either in the internal data register or in the internal data direction register. If a PIA pin it setup as an input, then pull-up resistors in the Atari pull it "high". If a PIA pin is setup as an output, then the data register controls if a "low" or a "high" signal is output.

The CPLD emulates exactly this behavior and therefore behaves exactly like every other RAM extension in the Atari. The Oldrunner OS for example sets `PORTB` up as input because the 400/800 had 4 joystick ports – but no RAM/ROM control via `PORTB`. This automatically deactivates every RAM extension, because bit 4 of `PORTB` must be "low" to activate the RAM extension. But the emulated pull-up resistor in the CPLD pulls the pin "high" automatically in this case, like it would have been the case in the PIA.

The rest of the RAM extension logic is relatively simple. If the "**Ramdisk**" switch is turned `ON` and if bit 4 of the emulated `PORTB` register has the value `0`, then the internal memory of the Atari is disabled for every access to $4000...$7FFF and a 16k bank of the 512k RAM extension is mirrored in instead. The bits 2,3,5,6,7 of the emulated `PORTB` register determine in this case which of the 32 banks of 16k size shall be used.

8.5 Oldrunner

The Oldrunner is active if the "**OldOS**" switch is turned to `ON`). For every access to $E000...$FFFF bank 127 of the flash ROM is mirrored in and for every access to $C000...$CFFF the internal memory is deactivated. This makes the Atari behave exactly like the original Atari 400/800, which did not have any memory at $C000...$CFFF. When the Math Pack area ($D800...$DFFF) is accessed, the internal OS ROM of the Atari will be mirrored in and the Math Pack contained in the OS will be used.

8.6 Cartridge emulation

The cartridge emulation offers very many configuration options. Controlling all these options only with switches would not have been handy. Therefore the major part of the configuration is performed via software, for example from within the menu of the cartridge emulation or from within the debugger of the freezer.

8.6.1 Configuration switches

One function of the "**CartEmu**" and "**FlashWrite**" switches is to control access to the configuration registers. If both switches are turned `OFF`, then the configuration registers are inaccessible. If at least one of the switches is turned `ON`, then the configuration registers $D590...$D596 can be accessed.

If the "**CartEmu**" switch is turned `ON` when the Atari is switched on, then bit 2 of the register $D595 is initialized to `1`. This disables the rest of the cartridge emulation configuration and activates the cartridge emulation menu.

If the "**FlashWrite**" switch is turned `OFF`, then no write access to the flash ROM resp. the freezer RAM is possible. The only exception is the `8k+RAM` mode in which write access to the optional 8k freezer RAM bank at $8000...$9FFF is always possible. If the "**FlashWrite**" is turned `ON`, then write access to the flash ROM and the freezer RAM is possible after is has been enabled via bit 0 of the register $D595.

8.6.2 Configuration registers

The configuration registers at $D590...$D596 become accessible if at least one of the "**CartEmu**" or "**Flash Write**" switches is turned `ON`. These registers are writeable and readable. All unused bits of these registers should be set to `0`. During a read access to the registers, all unused bits also return `0`. All registers except register $D595 are set to $00 during power-on. If the "**CartEmu**" switch is turned `OFF`, the register is set to $00. If it is turned `ON`, the register is set to $04. Depending on the active modules type (e. g. in OSS or AtariMax mode) additional registers are accessible in the area $D500...$D5FF.

$D590: Bank (r/w)

7	6	5	4	3	2	1	0
-	bank						

Number of the selected bank for the cartridge emulation (0...127). Bit 0 is ignored in 16k mode.

$D591: Bank Enable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	enable

State of the ROM bank of the module. The value `0` disables the module, `1` enables it. In `8k+RAM` mode this bit only controls the area at $A000...$BFFF. The optional RAM area at $8000...$9FFF is controlled by the register $D593.

$D592: RAM Bank (r/w)

7	6	5	4	3	2	1	0
-	-	bank					

Number of the selected RAM bank (0...63) in `8k+RAM` mode.

$D593: RAM Bank Enable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	enable

State of the RAM bank of the module. In `8k+RAM` mode the value `0` disables the RAM bank, `1` enables it.

$D594: Mode (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	mode		

Type of the emulated module. The following modules types are supported.

- `0`: Cartridge emulation deactivated
- `1`: 8k module
- `2`: 8k+RAM module
- `3`: 8k module with TURBO FREEZER 2005 bank switching
- `4`: 16k module
- `5`: OSS module
- `6`: AtariMax module
- `7`: Reserved, do not use!

$D595: Misc Configuration (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	menu	source	write

`menu` Cartridge emulation menu control
- `0` Normal cartridge emulation
- `1` Activate cartridge emulation menu (bank 126)

`source` Module data source
- `0` Flash ROM
- `1` Freezer RAM

`write` Write access control
- `0` Write access disabled
- `1` Write access allowed, if the "**FlashWrite**" switch is turned `ON` in addition

$D596: SDX Configuration (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	config	

`00` SpartaDOS X deactivated

`01` SpartaDOS X in freezer RAM starting at bank 0

`10` SpartaDOS X in flash ROM starting at bank 0

`11` SpartaDOS X in flash ROM starting at bank 64

8.6.2.1 Additional registers in 8k mode with TURBO FREEZER 2005 bank switching

$D540...$D57F: Bank Select (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this area selects the bank. An access to address $D540 selects bank 0, $D541 selects bank 1, ..., $D57F selects bank 63. If the bank register $D590 is set to a bank in the range 64...127, then $D540...$D57F will refer to the banks 64...127 instead of the banks 0...63.

$D580: Cart Disable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this register disables the module.

$D581: Cart Enable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this register enables the module.

$D584: Write Disable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this register disables write access to the module.

$D585: Write Enable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this register enables write access to the module.

8.6.2.2 Additional registers in OSS mode

$D500...$D50F: OSS Bank Select (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

These registers implement the bank select and cart disable as in OSS modules of type "M091". A read or write access to the respective address activates the following configuration.

$D500 Module enabled, $A000...$AFFF bank 1, $B000...$BFFF bank 0

$D501 Module enabled, $A000...$AFFF bank 3, $B000...$BFFF bank 0

$D508 Module disabled

$D509 Module enabled, $A000...$AFFF bank 2, $B000...$BFFF bank 0

8.6.2.3 Additional registers in AtariMax mode

$D500...$D57F: Bank Select (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this area activates the module and selects the bank. An access to address $D500 selects bank 0, $D501 selects bank 1, ..., $D57F selects bank 127.

$D580: Cart Disable (r/w)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

A read or write access to this register disables the module.

Caution: In the original AtariMax module the addresses $D581...$D5FF can also be used to disable the module. The cartridge emulation only offers this single address for this.

8.6.2.4 Additional registers when SpartaDOS X is active

$D5E0: Bank Select (r/w)

7	6	5	4	3	2	1	0
control		bank					

`control` SpartaDOS X control
- `00` SpartaDOS X enabled, additional module disabled
- `01` SpartaDOS X enabled, additional module disabled
- `10` SpartaDOS X disabled, additional module enabled
- `11` SpartaDOS X disabled, additional module disabled

`bank` SpartaDOS X bank select (bank 0...63)

If the SpartaDOS X emulation has been activated via the register $D596, it has priority over the rest of the cartridge emulation. The cartridge emulation then behaves as if an original SpartaDOS X module was "plugged in".

9 Further information and downloads

- TURBO FREEZER
  `http://turbofreezer.horus.com`
  Up to date information, updates and more.
- ABBUC
  `http://www.abbuc.de`
  Source of supply where TURBO FREEZER can be ordered. This website is mainly in German language.
- Xilinx
  `http://www.xilinx.com`
  Data sheets of the CPLD XC95144XL, download of the development environment ISE WebPACK.
- ST Microelectronics
  `http://www.st.com`
  Data sheets of the Flash ROM 29F040B.

Warranty and intended usage

The warranty on material and assembly is 6 months. If the freezer is defective, it may be replaced by one that is functioning. In case of hardware incompatibility, the price of the freezer will be refunded, after the part has been returned, free of postage. This warranty only covers the hardware. The risk for error free operation of the software and the application for specific tasks rest solely with the buyer or user. There are no rights, directly or implied, of whatever form, concerning software, exchange, improvements, refunds (fully or partially), changes, updating etc. Warranty does not cover damage caused by transport, incorrect installation or use, static discharges, pollution, attempted extensions and any alterations to the electronics outside our workshop. There's also no warranty on any parts that were not installed by us.

Usage of the TURBO FREEZER must not in any way infringe rights of third parties or violate any laws or regulations. Any responsibility or claims by third parties of any form, illegal use, damage to property or dead or living matter, and any other kind of claim is not accepted by us in any way and are the sole responsibility of the user. It's the sole responsibility of the user, owner or buyer to prevent illegal use or any other damage that may arise from using the product. Adults are responsible for their children. The user should be aware of regulations or laws concerning the usage and seek council in case of doubt. Claiming ignorance or lack of knowledge is not an option out.

The use of commercial names, trademarks, labels or manufacturer makes is not done with the assumption that any of these items should be freely usable and no rights of the items are intentionally claimed or assumed. All references do not take any patents or other rights into account. Any claim from manufacturers is strictly excluded. Use at your own risk.