Action Replay 5

1. Introduction

Action Replay 5 is an update to the Action Replay Cartridge developed by Datel Electronics for the Commodore Amiga. The original device was developed by Olaf Boehm and Joerg Zanger and sold as Action Replay, Action Replay II and finally Action Replay III. This device was designed to be used on the Commodore Amiga A500 and a version was also released for the A2000. The code was later extracted from the ROM and updated by Michael Pendec (BlackHawk/Paradox) and released as a software debugger called ARIV for the Amiga A1200/4000.

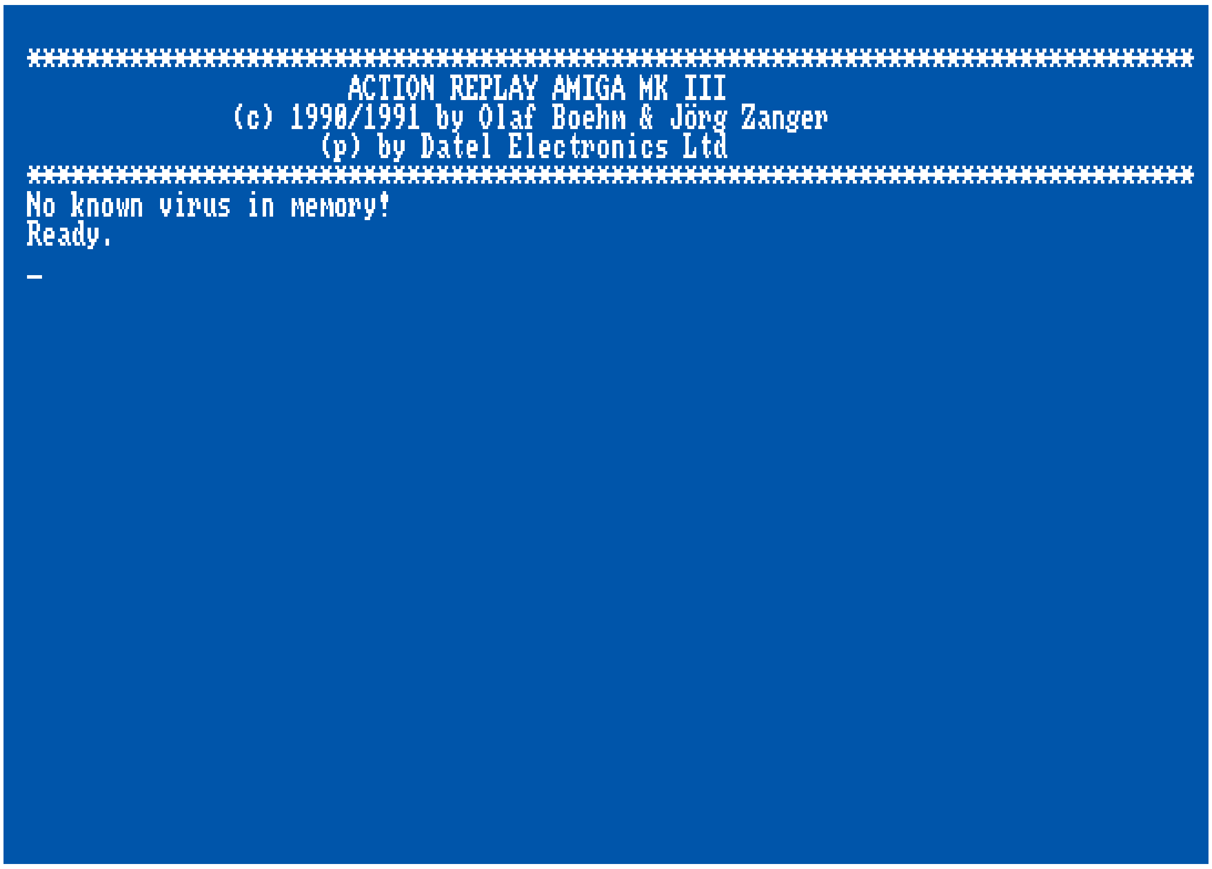
This new release is based upon the code in both the Action Replay III cartridge and the software ARIV release. It combines all of the features from these but also has many new features and bug fixes. Here is a brief list of some of the new features:

- UK and Italian keymaps
- Ability to save preferences to flash memory
- Enhanced disk read/write routines
- Rob Northen Copylock finder
- Replaced Burst Nibbler with XCopy
- Utilisation of the full PAL screen area
- Ability to deploy Rob Northen DISKIO/DOSIO routines into memory
- Send and receive files and data via serial port
- AmigaXfer support

This new Action Replay can run in a number of different configurations (which are further explored below) and while the firmware is backwardly compatible with the original Action Replay MK3 and with WinUAE the full feature set is only available when used with the DeMoN hardware.

Original Hardware - Action Replay III

Na103 successfully reverse engineered the functionality of the Action Replay III and has provided instructions on how to build your own version of the hardware here https://github.com/na103/ar3. In addition he also reversed the A2000 version of Action Replay III and that is available here https://github.com/na103/bbar3. Action Replay 5 is fully compatible with both of these hardware designs - or if you already have an Action Replay and are skilled with soldering you can desolder the ROMs and replace them with new ones containing Action Replay 5. There are a small number of new features that will not be available using this hardware but there are many others that will work fine.



```
********************************************************************************
                      ACTION REPLAY AMIGA MK III
              (c) 1990/1991 by Olaf Boehm & Jörg Zanger
                    (p) by Datel Electronics Ltd
********************************************************************************
No known virus in memory!
Ready.
_
```

The original Action Replay MK3

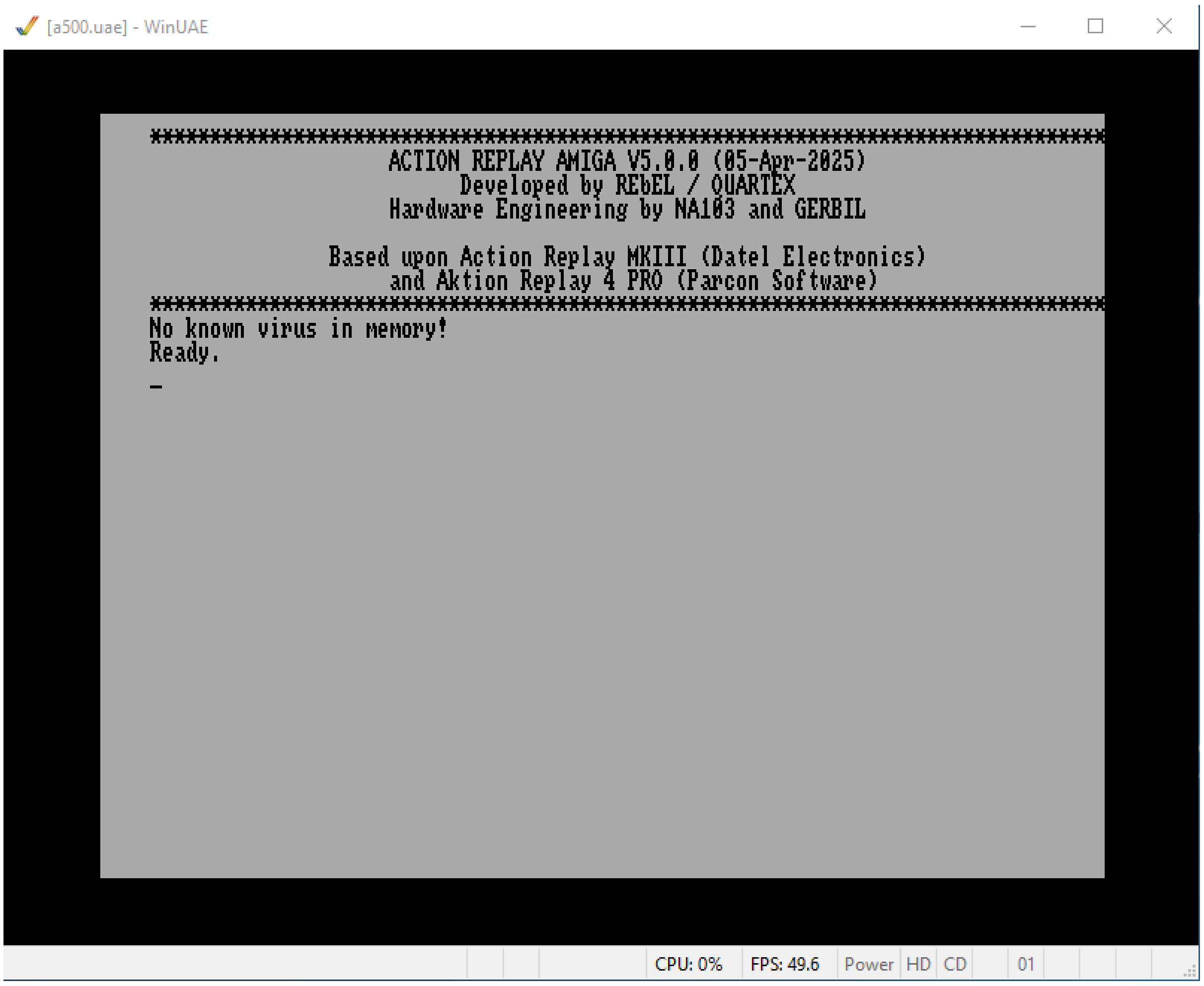
WinUAE - Action Replay Emulation

The Action Replay 5 firmware has been designed in a way that it can also run under WinUAE and will function similarly to running it on original hardware. However the emulation of Action Replay under WinUAE is not completely accurate and in fact it emulates slightly more RAM in the cartridge than was actually present.

The original Action Replay hardware only contained 40k of RAM but it was provided by two chips. The first was a 32k chip and the second was an 8k chip. Due to the way these were arranged it would have been easier for WinUAE just to emulate a full 64k of RAM rather than emulating the exact memory configuration of the Action Replay III.

With Action Replay 5 we detect this extra memory and we use the extra RAM to allow the Action Replay console to display 31 lines of text instead of being limited to the 25 line NTSC screen area (obviously if the user is running in NTSC screen mode this does not apply).

When run under Winuae the Action Replay 5 firmware is compatible with many more configurations than the original hardware was. It will run on any Amiga model with any 680x0 cpu and has also been tested under Kickstart versions 1.2, 1.3, 2.0, 3.0, 3.1 and 3.2. Some of the features however, only work on earlier Kickstart versions 1.2 and 1.3 (particularly the memory management features).



```
*******************************************************************************
                ACTION REPLAY AMIGA V5.0.0 (05-Apr-2025)
                     Developed by REbEL / QUARTEX
                 Hardware Engineering by NA103 and GERBIL

             Based upon Action Replay MKIII (Datel Electronics)
                 and Aktion Replay 4 PRO (Parcon Software)
*******************************************************************************
No known virus in memory!
Ready.
_
```

Action Replay 5 running in WinUAE

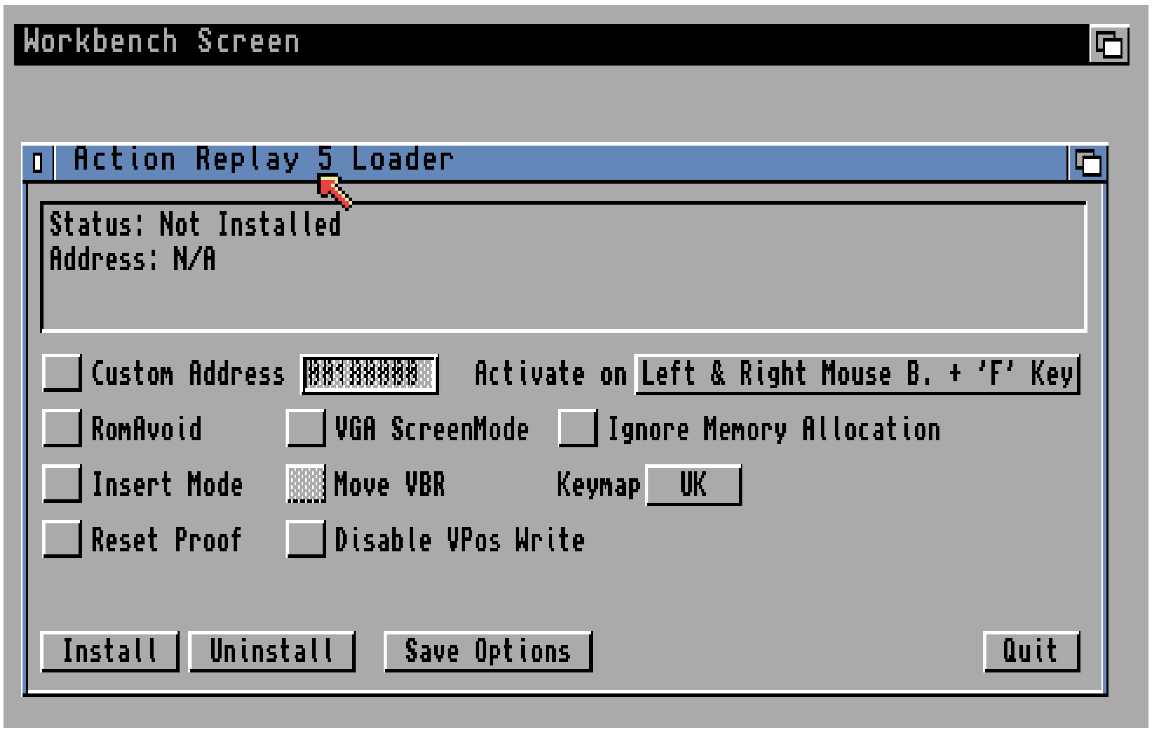
Software Only Mode

As with the ARIV predecessor, Action Replay 5 can also run without any additional hardware. There is a software loader included in the archive that is capable of loading Action Replay 5 into an area of memory and it then takes over several interrupt vectors in order to provide a trigger hook to allow you to launch into the Action Replay console via keyboard or mouse presses. The software only version also supports the use of an NMI switch if you have fitted one to your Amiga.

This software version of Action replay requires around 350k of RAM to load itself into and you can control where it is loaded if you wish. When running the software version, the software can be disabled if the software you are running on the Amiga overwrites the interrupt vectors that Action Replay 5 intercepts or if the interrupts are disabled. This is not possible to completely prevent but there is an option on the loader interface to move the VBR (If you have a processor that allows this). This will mean older software that does not know about the VBR present on newer CPUs will be less likely to disable the software.

All of the features of Action Replay III that were removed to make the ARIV version of Action Replay have been reinstated in this version, so that does mean it needs more memory. All of the features and changes that were made in ARIV have also been retained so this version really is the best of both worlds.

This software only version is useful if you want to access the Action Replay and do not have the necessary hardware or if you have a machine that cannot run the hardware versions. It is not however as fully capable as the hardware versions and can more easily be detected by software and may require the user to take additional steps to prevent the code being debugged from disabling or overwriting the software.



Status: Not Installed
Address: N/A

Custom Address 00100000
Activate on Left & Right Mouse B. + 'F' Key
Keymap UK

Action Replay Software Mode Loader

DeMoN Cart - Our Enhanced Action Replay Cart Remake

This new hardware is the brainchild of Na103, Gerbil and Rebel. The name DeMon actually comes from the initials of our real names. If you want to take full advantage of the new features of Action Replay 5 - this is the way to go. It has additional memory that is available for the Action Replay firmware to use and it also has its firmware stored on flash chips which means the Action Replay 5 firmware can be upgraded easily. Details for building your own DeMon cart are here https://github.com/gerbilbyte/DeMoN

The extra memory (1mb in total) in the DeMoN is only usable from the Action Replay console (since like the original cart the memory is hidden when the cart is not active). However it can be used as an 880k ramdisk when in the console and will retain its contents on a soft reset.

This cart has a feature where the stealth feature of the hardware can be disabled so that the Action Replay memory can be visible even when not in the Action Replay console. This is useful if flashing fails for any reason and you need to run the stand-alone flasher to unbrick your device.

Due to the fact that the firmware now exists on a flash memory chip rather than a ROM chip, when using the DeMoN it is also possible to save your preferred settings back to the chip and these settings will be retained even after power down and they will also be transferred to the new firmware when it is flashed.

Software Failure. Press left mouse button to continue.
Guru Meditation #81038001.48454C50
DeMoN
The DeMoN cart

2. Getting Started - The Action Replay 5 Console

Upon pressing the button on the Action Replay Device (or triggering the software with the relevant key/mouse combination) the running program will be frozen and you will be taken to the Action Replay console. From here you can use any of the built in commands to perform any actions you wish before exiting back to the running program.

Some of the functions of the Action Replay are:

- File/Disk management
- Debugger/Monitor
- Disk Copying
- Music/Sample Ripping
- Graphics Ripping
- Trainer
- Program Freezing
- System Information

When at the command prompt in the Action Replay console you can press the help key and a complete help text will be displayed. The help text is several pages long and includes a list of all of the commands.

In addition to this a new feature of Action Replay 5 is that you can enter any command followed by a “?” to display the syntax for that particular command.

In addition the following keys are also defined for use in the console.

- Shift - Pause scrolling
- Shift Help - Show command aliases/shortcuts
- Tab - Insert spaces
- Escape - Cancel current command
- F1 - Clear the screen
- Shift F1 - Cursor home
- F2 - Restore from second screen
- Shift F2 - Save to second screen
- F3 - Edit preferences
- F4 - Repeat last command
- F5 - Send current screen t printer
- F6 - Toggle printer dump on/off
- F7 - Switch between insert/overwrite modes
- F8 - Show mempeeker keyboard shortcuts
- F9 - Cycle between keyboard layouts (UK/DE/USA/IT)
- Shift F9 - Compare screen pages
- F10 - Switch screens
- F11 - Switch between 15Mhz/31Mhz (AGA Chipset only)

3. The Preferences Editor

Pressing F3 at the console command prompt will bring up the preferences editor. Here you can reconfigure a number of user settings. These settings will be retained until the system is powered down or if you have a DeMoN cart you can save them to the flash memory and they will be retained permanently. You can also save the preferences to disk and reload at a later date if required.

The preferences screen consists of two pages of options. Some of these options may be disabled if not supported in your version of the Amiga operating system (a number of the options only work on Kickstart versions 1.2 and 1.3). The preferences are edited using the mouse and pressing ESC will exit the preferences editor.

<u>Memory Control</u> (KS 1.2/1.3 only)

The system memory is managed using these settings. Extra Chip and Fast memory can be enabled/disabled here. If disabled these areas will still be visible in the Action Replay console but will not be available to the Amiga operating system. A reset is needed to apply changes made here.

<u>Module Interna</u>

These options affect the internal workings of the Action Replay. The NoRes option disables much of the functions executed on a reboot (including the reset logo).

The test1 and test2 options are different types of system reconfiguration when using the X command to restart the machine after freezing, in some cases the machine can lock up. If this happens you should try setting the test1 and/or the test2 options (information taken from original Action Replay 3 manual)

The blanker option enables/disables the screen saver. The screen will be blanked if no activity is detected in the console and you can press any key to turn the display back on.

The bottom left section is used to select the colours that the Action Replay console is displayed in. The default is the new grey ARIV version but you can change it back to the old blue and white style if you prefer.

<u>Megastick</u>
The megastick option is for the joystick translation codes which can be entered using the megastick command. The two meters on the right of the screen are used to allocate an auto fire rate for the two joysticks. They can be set totally independently of one another so a player can be handicapped.

<u>Autoconfig</u>
The autoconfig option allows Action Replay to detect what it thinks the computer is, for example when on it will detect and boot an A590 hard drive. This will also disable autoconfig fast memory when the option is disabled.

Boot Selector (KS 1.2/1.3)
The boot selector in the top left is used so you can get the machine to boot from any drive. You may select any drive available or variable so that the computer will search each drive for a bootable disk. With this option there are two things you should bear in mind. The first is that when you have selected this option and exited you must reboot the machine before it will work even if you are on the kickstart screen. The second thing is that the success of this option depends on the disk you are booting from, i.e. if it starts to boot from disk df1: and the program tells the computer to go and read from df0: the boot will fail, so please bear this in mind. The easiest way to try out the variable boot is to use an Action Replay disk that has a bootblock on it (using the install command).

Bootblock Coder (KS 1.2/1.3/2.0)
This allows the bootblock to be encoded with a key that will prevent the disk from booting without the relevant key. It will also mean that the disk cannot be booted without the hardware cartridge.

Disk Coder
The disk coding section is now available from the second preference screen. Each option can either be enabled or disabled by clicking on the appropriate icon. For more information on these features please see the section on disk based instructions

Drive Control (KS 1.2/1.3 only)
Drives can be turned off or on by clicking on the appropriate boxes.

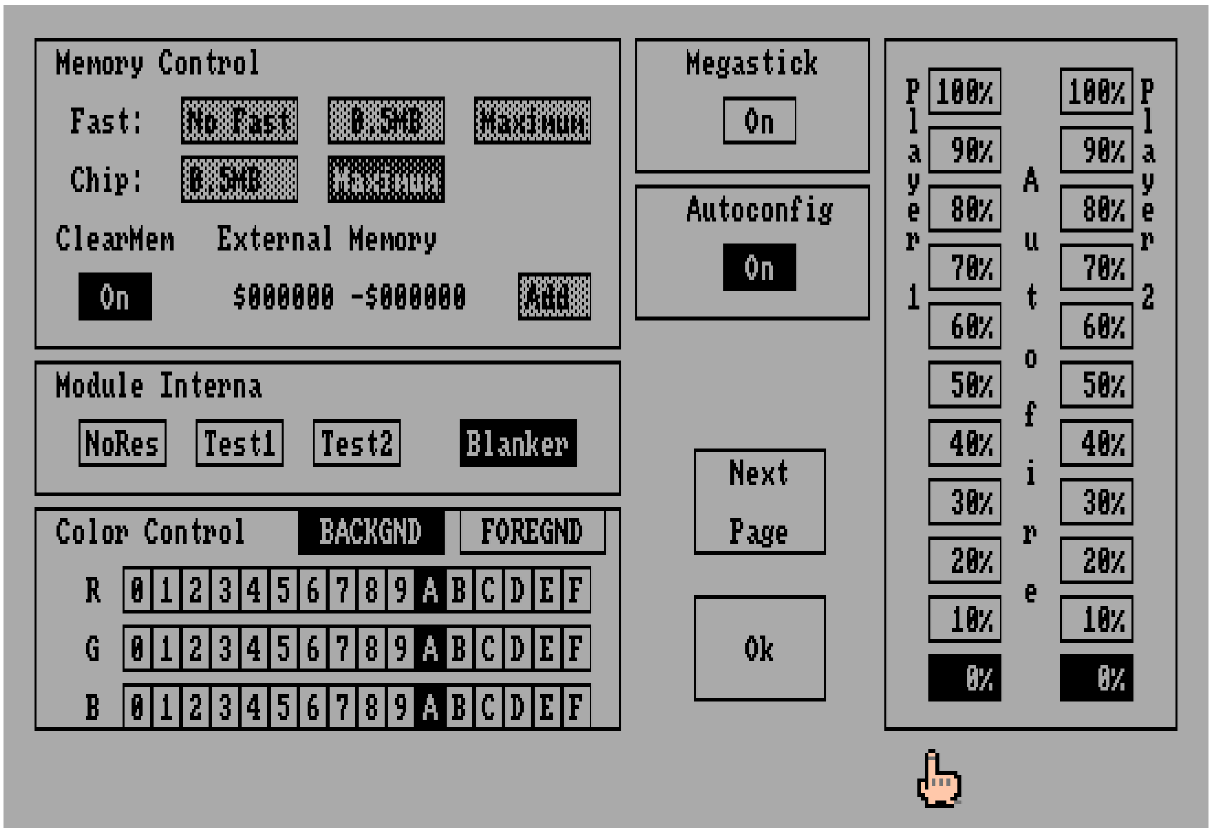
Virus Test
Using these options you can switch off the automatic virus detection. This is useful when you are using disks that have built in virus protection, e.g. Sentinel, which contain parts of the viruses so they can be identified. This code is then interpreted by Action Replay as a virus itself. Kill is useful for similar reasons. You would normally leave it active so a virus that is found is killed. Virus boot will check the bootblock of your disk as well as checking for a virus in memory. This option should be turned off if you are using a hard drive.

XCopy
The X-Copy faststart option allows entry to X-Copy on a reset by pressing the left mouse button and holding it down.

Save/Load
There is now an option to save and load your Action Replay preferences to a floppy disk; simply click on the save/load option on the second screen and select an appropriate path/filename.

Setmap (KS 1.2/1.3 only)
The setmap D function is for German users and sets the keymap to their German standard (e.g. Z and Y are switched)

Safedisk (KS 1.2/1.3 only)
There are two options here. Resident cures the ROM bugs present in the Amiga which can cause disk failure and the No Click option prevents the floppy disk drive from clicking. For further info see the Safedisk command.


Memory Control
Fast: No Fast | 0.5MB | Maximum
Chip: 0.5MB | Maximum
ClearMem [✓ On]
External Memory $000000 -$000000 Add

Megastick: On
Autoconfig: [✓ On]

Module Interna: NoRes | Test1 | Test2 | [✓ Blanker]

Color Control: [✓ BACKGND] | FOREGND
R 0 1 2 3 4 5 6 7 8 9 [✓ A] B C D E F
G 0 1 2 3 4 5 6 7 8 9 [✓ A] B C D E F
B 0 1 2 3 4 5 6 7 8 9 [✓ A] B C D E F

Next Page
Ok

Player 1 Autofire: 100% 90% 80% 70% 60% 50% 40% 30% 20% 10% [✓ 0%]
Player 2 Autofire: 100% 90% 80% 70% 60% 50% 40% 30% 20% 10% [✓ 0%]

Action Replay 5 Preferences Screen

Things to remember when using Action Replay

Firstly, the default numbering system is hexadecimal. If you want to enter decimal numbers you must precede them with a ! symbol. If you need to enter binary numbers then the prefix is % and there is no need to prefix if you want to enter hex.

Secondly, when editing preferences, many of the options do not take effect until after a reset. Exit the preferences screen and do a warm reset using Ctrl-Amiga-Amiga and the options will take effect.

Finally, when entering commands at the console, file paths are specified slightly differently than in standard AmigaDOS format. Drive names are specified as simply 0: 1: 2: 3: rather than DF0: DF1: DF2: DF3: If you have a DeMoN cart there is also a built in ram disk that can be accessed as R:

The following sections document the actual commands available in the Action Replay console:

4. File/Disk management Commands

These commands allow common file and disk based tasks to be completed without needing a workbench disk. Many of the file based commands will operate on the current path or you can specify a complete path as part of the command.

CD (<path>)
Changes the current directory. You can use CD / to go up one level in the directory structure.

DIR (<path>)
This shows the contents of a directory. If no path is specified then the contents of the current directory will be shown.

DIRA (<path>)
Similar to DIR this shows the contents of a directory. With this command all sub-directories will also be disabled.

COPY (<path>)<source-file>,(<path>)(<dest-file>)
Copy a file from one location to another. The whole file needs to fit in the disk buffer so you may need to kill the running program if the file is large (an option will be given for this if required).

FCRC16 (<path>)<file>
This command calculates a 16 bit checksum from the file contents.

FCRC32 (<path>)<file>
This command is similar to the FCRC16 command above but it calculates a 32 bit checksum from the file contents.

TYPE (<path>)<file>
Displays the ascii contents of a file to the screen. If the file contains binary data you may wish to use the DUMP command instead to display it as hex/ascii.

DUMP (<path>)<file>
Displays the contents of a file as a hex/ascii dump (similar to the output of the memory dump command).

RENAME (<path>)<old-file>,<new-file>
Changes the filename of a file from the old-file to the new-file. Unlike the AmigaDos rename command, you may not move files with this command.

MAKEDIR <path>
This instruction will create a subdirectory at the point specified by (path). If no path is specified a new directory will be created in the current directory,

    e.g. MAKEDIR SUB1

will create the sub directory SUB1 in the current directory

    MAKEDIR MAIN/SECOND/SUB1

will create the sub-directory SUB1 in sub-directory SECOND in the main directory MAIN.

DELETE (<path>)<file>
Removes a file at the location specified. If no path is included the current directory is assumed.

FORMAT (<name>)(,FFS)
Formats the disk in the currently active drive. If no name is given a default name will be assigned. The FFS option will format the disk as FFS - the default is OFS. This is a full format where every track is initialised. Verify is not performed, please see the FORMATV command if you wish to verify.

FORMATQ (<name>)(,FFS)
Performs a quick format of the currently active drive. Only the necessary sections of the disk are re-initialised so the disk should have been previously fully formatted to use this option.

FORMATV (<name>)(,FFS)
Performs a full format of the disk and in addition performs a check to ensure that each track has formatted successfully.

INSTALL (<bootblock-nr>)
Installs a bootblock on the disk to create an autobooting disk. There are two types of bootblock that can be installed. The first (0) is the standard AmigaDos boot block and the second (1) is an anti-virus bootblock. The default is the standard AmigaDos boot block. It should be noted that the anti-virus bootblock is not compatible with Amiga OS 2 or higher.

DISKCHECK (<drive>)
Performs a track by track scan of the disk to check for errors. Any non-standard tracks will also be highlighted as errors. If the drive is not specified then the current drive is used.

DISKWIPE (<drive>)
Performs a quick wipe of the specified drive (or the current drive). Random data is written to the disk effectively unformatting it.

RELABEL <diskname>
Updates the label of the disk in the current drive to the name specified.

BOOTCODE (<codenumber>) (Hardware devices only)
Set the boot code to allow bootcode protected disks to be booted. This command is only compatible with Kickstart versions 1.x and 2.x - use of this tool is not recommended since once the bootblock has been encrypted it will not be possible to boot this disk on machines using newer kickstart versions. Setting the boot code can also be done from the preferences screen. If no parameter is entered it displays the current settings for all drives.

BOOTPROT <codenumber) (Hardware devices only)
This command protects the bootblock by encrypting the disk format id and the boot sector checksum. You can use the bootcode command (as above) to install a patch into the Amiga OS to allow the disk to be booted.

If you wish to remove the boot protection, you should re-apply the same code. Applying a second code onto an already protected disk is not recommended.

It should be noted that the bootcode command is not compatible with Kickstart versions 3.0 or higher.

CODE (<drive> <code-number>) (Hardware devices only)
This is another disk encryption tool that protects the whole disk rather than just the bootblock. Encryption codes can be set for each drive individually and the current state can be displayed using the code command by itself. If you lose the encryption code then your data will become unreadable so take care when using this command. Running this command with no parameters will display the current drive coding settings.

DCOPY <source-drive> <dest-drive>
Copy an entire AmigaDOS disk. If the source and destination are the same then the data will be copied into memory and written back to the disk. If the source and destination are not the same the disk will be copied one track at a time.

XCOPY
Launch X-Copy. This program will be copied to the Amiga chip RAM destroying the currently running program. There is no way to return to Action replay from the X-Copy. You must reset once you are done with the tool Attempting to exit X-Copy will result in a reset.. X-Copy is a fully featured disk copy program that many users will be familiar with and is capable of copying non-dos disks. The detailed usage of this tool is outside of the scope of this manual. This tool replaces Burst Nibbler that was present in Action Replay Mk3.

CODECOPY <source-drive> <dest-drive> (Hardware devices only)
Similar to the dcopy command. It copies an entire disk but it also applies the decoding/encoding set with the code command. It can therefore be used to encrypt or decrypt an entire disk.

SAFEDISK (a/b/s/n/u/v/q)
Applies patches to the system trackdisk device. The patches depend on the parameter passed in. This function only works on kickstart 1.x versions of the trackdisk device.

N - NoClick
B - Patch bugs in trackdisk that can cause files to be damaged
S - Attempt to read damaged tracks and ignore read errors
V - Verify writes
U - Update tracks. f there is no access to the disk for a short while, reset the track buffer and switch off the drive motor
A - All
Q - Turn off all options.

RT <start-track> (<num-tracks> <dest-addr>)
Read a number of tracks from an AmigaDOS disk directly into memory. If you do not specify the memory address to write to, a buffer is allocated.The start track number is specified as a number between 0 and !159 The tracks are read from the currently active drive.

RS <start-sector> (<num-sectors> <dest-addr>)
Read a number of sectors from an AmigaDOS disk directly into memory. If you do not specify the memory address to write to a buffer is allocated.The start sector number is specified as a number between 0 and !1759

RB <start-offset> (<num-bytes> <dest-addr>)
Read a number of bytes at any offset from an AmigaDOS disk directly into memory. If you do not specify the memory address to write to a buffer is allocated.The byte offset is specified as a number between 0 and !901119

RP <start-track> (<num-tracks> <dest-addr> <pdoskey>)
Similar to the RT command but this is used on disks formatted using the Rob Northen PDOS protection. This protection is often used on Team 17 games among others. PDOS disks have 12 sectors per track. PDOS disks also have an encryption key which can be specified. If it is not included the key will be automatically calculated.

RPB <start-offset> (<num-bytes> <dest-addr> <pdoskey>)
Similar to the RB command but this is used on disks formatted using the Rob Northen PDOS protection. PDOS disks have 12 sectors per track so the disk holds a maximum of 983040 bytes if all tracks are formatted using this system.

RPS <start-sector> (<num-sectors> <dest-addr> <pdoskey>)
Similar to the RS command but this is used on disks formatted using the Rob Northen PDOS protection. PDOS disks have 12 sectors per track so the maximum sector number is !1919

RR <start-track> (<num-tracks> <mfm-sync> <readlen> <dest-addr>)
Read raw mfm data from the disk with no decoding. The initial mfm sync value is copied to the memory output before it is not included in the length so the actual memory data will be 2 bytes longer than the specified readlen. This is done this way to make the decoding process (see MFM command) simpler and so that the track data is in a suitable format to be written back to disk using the WR command.

WT <start-track> <num-tracks> <src-addr>
Write a number of tracks to the active drive from memory. The tracks are numbered 0-!159. When using an allocated disk buffer, the source address can be specified using T to denote a track address, S to denote a sector address

WP <start-track> <num-tracks> <src-addr> <pdos-key>
Write a number of tracks to the active drive using PDOS format. You must specify a PDOS key with which the data will be encrypted. The PDOS format that is output is compatible with the official PDOS format but does not contain the sector gap(s) which decreases the size of the track so that it can fit when written using a standard drive.

WR <start-track> <num-tracks> <src-addr> <word-length>
Write raw tracks to the currently active drive. If the data was read using the RR command the write length will need to be 2 bytes longer than the read length to account for the sync word at the start of the data.

MFM <src-addr> <track-len> <track-count> <dest-addr> <sync> <sector-offset> <sector-count> <sector-len> <sector-interleave> (<mfm-type>) (<sectornum-offset>)
Decodes raw mfm data. This command can be used to decode data read into memory with the RR command. The command has many parameters which are used to describe how the decode should work. This is necessary to allow decoding of a wider range of mfm formats. Unfortunately the number of possibilities of custom disk formats does mean that the command will never be able to handle every possibility.

The parameters are:

src-addr: The address of the source mfm data.

track-len: The track length.

track-count: number of tracks to decode.

dest-addr: The destination address for the decoded data

Sync: This identifies the sync marker in use. Some custom formats have a single sync marker at the start of the track and others have sync markers at the start of each sector. The decoding will search for the specified sync marker at the start of each sector. If the format you wish to decode only has one sync marker at the start of the track you may be able to decode the tracks as if they were only 1 large sector or you may specify 0 as the sync marker to disable the searching for sync markers between sectors.

Sector-offset: This is the offset within the sector to the actual sector data. There may be sector header data in the custom format you are working with, so you can set this value to skip over those. This value should be the amount of bytes to skip in the mfm data after the sector sync (if the sector sync is repeated then the offset will be taken from after the last sync value).

Sector-count: This is the number of sectors you expect to find per track in the mfm data.

Sector-interleave: This defines the way in which the sector data is stored. The mfm data is stored in two blocks and these are combined together. Typically the data is stored either in a 4 byte interleaved configuration where 8 bytes are read and combined to give a 4 byte output (you should specify 4 for this configuration) and this is repeated across the whole sector or as per AmigaDOS format in a 512 byte interleave configuration where 512 bytes are read and combined with the second 512 bytes.

Mfm-type: This defines how the two sets of interleaved data are combined. It defaults to type 1 which is standard for AmigaDOS. Type 1 Is where the first interleaved data forms the higher bits in the output and Type 2 is where the second interleaved data is the higher bits.

So the mfm decoder for type 1 might look like this:

```
    MOVE.L (A0)+,D0       ;first 4 bytes of mfm data
    MOVE.L (A0)+,D1       ;second 4 bytes of mfm data
    AND.L #$55555555,D0   ;mask off clock bits
    AND.L #$55555555,D1   ;mask off clock buts
    LSL.L #1,D0           ;shift first bits into place
    OR.L D1,D0            ;or in the second bits
    MOVE.L D0,(A1)+       ;write the output
```

and the mfm decoder for type 2 might look like this:

```
    MOVE.L (A0)+,D0       ;first 4 bytes of mfm data
    MOVE.L (A0)+,D1       ;second 4 bytes of mfm data
    AND.L #$55555555,D0   ;mask off clock bits
    AND.L #$55555555,D1   ;mask off clock buts
    LSL.L #1,D1           ;shift second bits into place
    OR.L D1,D0            ;or in the second bits
    MOVE.L D0,(A1)+       ;write the output
```

Sector-num-offset: This parameter allows you to decode formats where the sectors are not necessarily in a fixed order, or where there is no track sync marker but only a sector sync marker. Decoding this data would not work correctly without this parameter set since the track in memory might start at any sector. This parameter defines the offset from the sector sync where the sector number is stored. The data is expected to be formatted as a 4 byte interleaved value so the offset should be to the byte containing the first half of the sector number and the second half will be assumed to be 4 bytes later.

For example in the case of AmigaDos format data the data following the sector sync is:

one byte of format byte (Amiga 1.0 format = $FF)
one byte of track number
one byte of sector number
one byte of sectors until end of write

So in this case the value for the sector offset would be 2.

Some examples of decoding of custom track formats may explain how this works:

Turrican 2 - Track length 9aa4, Sync 9521 (Treats the data as one big sector)
```
rr 2 1 9521 1aa6 40000
mfm 40000 1aa6 1 30000 9521 2 1 1a90 4 1
```

Zool - Sync 4489, length 180c (Similar to the above on but using type 2)
```
rr 3 1 4489 180e 40000
mfm 40000 180e 1 30000 4489 2 1 1800 2 2
```

Lemmings Sync 4489, length 1867 (Here we decode 4 sectors of $400 bytes but without using sector syncs, we simply set the offset to skip the sync)
```
rr 2 1 4489 1868 40000
mfm 40006 1868 1 30000 0 4 4 400 2
```

RNC
Reads a Rob Northen serial track from the currently active drive and attempts to display the possible values for the serial key. The exact key in use will depend on the code. You can use the FC command to attempt to locate the copylock code or CI to display the copylock info if you already know the address of the copylock code.

DMON
Displays the current disk monitor buffer address and size. If none is allocated then a buffer will be allocated.

CLRDMON
Clear the disk monitor buffer and reset the memory contents.

BAMCHK <addr>
Calculate bitmap allocation block checksum. Used to correct the checksum of a bitmap allocation block that has been read into memory before writing it back to disk.

BOOTCHK <addr>
Calculate bootblock checksum. Used to correct the checksum of the bootblock that has been read into memory before writing it back to disk. The bootblock is actually 2 sectors at the start of the disk (sector 0 and 1).

DATACHK <addr>
Calculate data block checksum. Used to correct the checksum of a data block that has been read into memory before writing it back to disk.

5. Debugger/Monitor Commands

SETEXCEPT
Sets an exception handler to trap any of the following exceptions and drop you at the Action Replay console if any of them are triggered:

- Address error
- Illegal instruction
- L’ine-A
- Line-F
- Division by zero

SETAPI
Initialises an API handler that can be called from user code via the TRAP #7 instruction. The value in the register D0 defines the operation and further details on the api calls can be found in Chapter 13

CLRAPI
Removes the API handler.

COMP <start-addr> <end-addr> <dest-addr>
Compare a block of memory against a second area of memory. The area between the start and end will be compared against the area starting at the destination. All differences in the destination block will be reported.

LM (<path>)<file>,<dest-addr>
Loads a file directly into memory.

SM (<path>)<name>,<start-addr> <end-addr>
Save a memory block directly to disk. The memory between the start and end will be saved to the file specified.

SMDATA (<path>)<file>,<start-addr> <end-addr>
Save a memory block as data. The memory between start and end is converted to data statements and saved to the file specified.

SMDC (<path>)<file>,<start-addr> <end-addr>
Save a memory block as DC.B. This works the same way as SMDATA but the data is generated as DC.B statements which are used by Devpac and other assemblers.

A <address>
Begin assembling 68000 code. The Action Replay changes to Assembler mode and assembly language commands can be entered which are assembled starting at the address specified. Press escape when you wish to return to the normal command mode.

B
This shows any breakpoints currently defined.

BS <addr>
Set a new breakpoint at a given address. When program execution hits this address the Action Replay will be activated. The memory at the given address must be writable as the Action Replay installs a TRAP instruction at this address in place of the normal opcode. You may add up to 5 breakpoints at any one time.

BD <addr>
Remove a breakpoint from a given address.

BDA
This removes all current breakpoints.

MS <address>
Sets a memory watchpoint at the specified address. When memory watchpoints are enabled the code will be run in the 68000 trace mode one instruction at a time. Once a command has been executed that changes the memory watchpoint the Action Replay will be activated. One limitation of using this command is that you cannot single step using the debugging at the same time as using memory watchpoints.

MW
Show all currently defined memory watchpoints.

MD <address>
Remove a memory watchpoint from the specified address.

MDA
Delete all currently active memory watchpoints.

ST (<steps>)
Using the 68000 trace function, run a specified number of steps. The ST command will trace into subroutines. If the step count is not included it will step one single instruction.

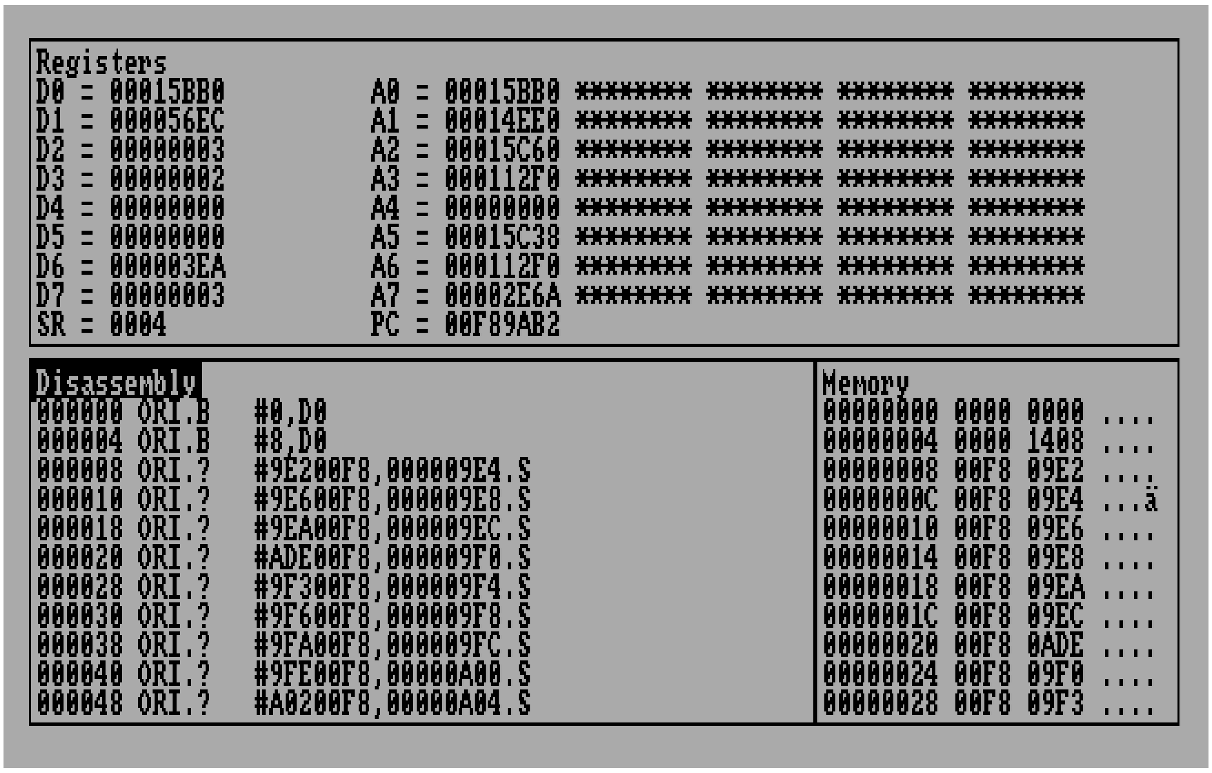
TR (<steps>)
Similarly to the ST command it executes code in single stepping mode. The TR command differs from ST in that it automatically traces over subroutines rather than tracing into them.

X
This exits from the Action Replay console and continues program execution at the current position.

C <1|2|address>
Disassembles the instructions that make up the copper list. You can use C 1 or C 2 to disassemble based on the current values of the cop1lc and cop2lc. If you pass in an address instead then the copper disassembly will begin at that address.

DBG
Launch into the interactive debugger. This is a useful tool when you want to step through some code while viewing memory areas at the same time. This debugger appearance will be familiar to users of Monam - the debugger that comes with the Devpac system. The following keys can be used in this mode:

- Escape - move the code window to the current position
- Cursor keys, move the current address
- M set the memory dump window location
- D set the code disassembly window location
- F1 switch between windows
- F5 resume executing code
- F7 step over subroutine
- F8 step into subroutine
- F10 return to the Action replay console



```
Registers
D0 = 00015BB0        A0 = 00015BB0 ******** ******** ******** ********
D1 = 000056EC        A1 = 00014EE0 ******** ******** ******** ********
D2 = 00000003        A2 = 00015C60 ******** ******** ******** ********
D3 = 00000002        A3 = 000112F0 ******** ******** ******** ********
D4 = 00000000        A4 = 00000000 ******** ******** ******** ********
D5 = 00000000        A5 = 00015C38 ******** ******** ******** ********
D6 = 000003EA        A6 = 000112F0 ******** ******** ******** ********
D7 = 00000003        A7 = 00002E6A ******** ******** ******** ********
SR = 0004            PC = 00F89AB2

Disassembly                                 Memory
000000 ORI.B   #0,D0                        00000000 0000 0000 ....
000004 ORI.B   #8,D0                        00000004 0000 1408 ....
000008 ORI.?   #9E200F8,000009E4.S          00000008 00F8 09E2 ....
000010 ORI.?   #9E600F8,000009E8.S          0000000C 00F8 09E4 ...ä
000018 ORI.?   #9EA00F8,000009EC.S          00000010 00F8 09E6 ....
000020 ORI.?   #ADE00F8,000009F0.S          00000014 00F8 09E8 ....
000028 ORI.?   #9F300F8,000009F4.S          00000018 00F8 09EA ....
000030 ORI.?   #9F600F8,000009F8.S          0000001C 00F8 09EC ....
000038 ORI.?   #9FA00F8,000009FC.S          00000020 00F8 0ADE ....
000040 ORI.?   #9FE00F8,00000A00.S          00000024 00F8 09F0 ....
000048 ORI.?   #A0200F8,00000A04.S          00000028 00F8 09F3 ....
```

Action Replay 5 Built in Debugger

D (<addr>)
Enter disassembly mode, disassembles code at the specified address one line at time. Pressing return will disassemble the next line of code. Escape will return to normal command mode. If no address is given then the disassembly will begin at the current program counter. Subsequent D commands will continue from where the last disassembly ended.

DD (<addr>)
Enters disassembly mode but initially displays 8 lines of disassembly. Further lines will be displayed upon pressing return.

DDD (<addr>)
Enters disassembly mode but initially displays 16 lines of disassembly. The disassembly will continue when return is pressed. Escape will return to command mode.

E (<offset>)
This displays the value of the custom chip register at the offset specified. You may edit the value shown and pressing return will update the value and display the next chip register entry. You can just press return without making any changes if you just wish to see the next value. Pressing Escape will return to command mode. If no offset is given the list will begin at offset 0.

EA
On AGA machines there are 256 colour palette entries and each one is 24 bits in size. The way this is done is by bank switching the original 32 palette registers. This means it is not possible to view all of the palette values using the E command. This command will dump the whole list of the 256 colour palette values in 24bit. If AGA is not detected this command will give an error. Press any key after the first page of values has been displayed to scroll to the remaining values.

F <string>(,<start-addr> <end-addr>)
Search for a string in a given memory range. If no memory range is given then the search will cover the memory areas known to the Action Replay (chip ram, slow ram and auto-config fast ram). This search is case sensitive.

FS <string>(,<start-addr> <end-addr>)
Search for a string in memory using a case insensitive search. See also the F command.

FA <addr> (<start-addr> <end-addr>)
Search in memory for an instruction that accesses a given address. It will also find branches and jumps to the address specified but it does not find a move into a data register with the corresponding address.

FAQ <addr> (<start-addr> <end-addr>)
Similar to FA but performs a quick search.

FR <string>(,<start-addr> <end-addr>)
Searches through memory for a relative string and displays the offset. For example if a piece of text is hidden by adding one character it will be shown as occurring with an offset of one so a search for “HELLO” on finding “IFMMP” will display an offset of one.

FC (<start-addr <end-addr>)
Searches for the start of a Rob Northen Copylock code.This searches for a particular instruction that is common at the start of many Copylock codes. If a possible Copylock routine is found then it also attempts to decrypt the copylock and display the type. The type can be used alongside the RNC command to identify the serial number.

CI <addr>
If you have manually identified the start of a Rob Northen copylock routine the CI command will attempt to decrypt the code and display the type information for the routine.

G (<addr>)
Resume program execution. If an address is specified then execution will resume at that address. If no address is specified then this command performs the same as the X command.

GK (<addr>)
This command resumes program execution (optionally at a given address) but prior to this it disables all interrupts, disables all DMA and pending interrupt requests.

TRANS <start-addr> <end-addr> <dest-addr>
Copy a memory block from one location to another. The memory between the start and end addresses is copied to the destination address. The transfer is capable of handling overlapping areas of memory.

WS <string>, <start-addr>
This command writes a string into a specific location in memory. In addition you may also specify a list of byte values rather than a string.

M <address>
Display/Edit memory as hex/ascii. One line of memory dump will be displayed starting at the address specified. The hex values can be edited and will be updated in memory when return is pressed. This will also move onto the next line of data. Pressing Escape will return to normal command mode.

MM <address>
Display/Edit memory as hex. This is the same as the M command but initially 8 lines of memory data will be displayed.

MMM <address>
Display/Edit memory as hex. This is the same as the M command but initially 16 lines of memory data will be displayed.

MEMCODE <start-addr> <end-addr> <code>
This command encodes an area of memory by EORing it with a byte value. Each memory location between the start and end addresses will be EOR.B'd with the specified code.

ADD <start-address> <end-address> <value>
Similar to the memcode instruction but rather than EORing the bytes a fixed value is added to each byte in the memory range between the start and end addresses.

N <address>
Show/Edit memory as ASCII. This works similarly to the M command but only ASCII data is displayed. The ASCII data can be edited on each line and pressing return will save the changes to memory and display the next line of ASCII. Escape will return to command mode.

NN <address>
Similar to N but 8 lines of text are displayed when the command is executed.

NNN <address>
As with NN but 16 lines are initially displayed.

NO (<offset>)
When a value is entered as (offset) any further use of the N command will add the (offset) value to the characters in memory that are displayed. Useful for displaying text that has been obfuscated by a simple addition or subtraction. NO by itself will display the current setting.

MQ <address>
Performs a quick dump of memory starting at the address specified. The memory dump will continue scrolling through memory until the Escape key is pressed.

NQ <address>
This command displays an ASCII dump of memory starting at the specified address. Only valid ascii characters will be shown and the dump will continue scrolling through memory intil the Escape key is pressed.

O <string>, <start-addr> <end-addr>
This command fills a memory area denoted by the start and end addresses with the string or bytes specified.

ROBD
Enables the Rob Northen decryptor. This function allows the decryption of a copylock routine to be bypassed displaying a disassembly of the decrypted code block. Running the robd command again a second time will disable the function.

R (<reg> <value>)
Show or update the main CPU registers present in the 68000 cpu. Entering R by itself will display all of the registers. Specifying the register and a new value will update the register with that value.

RC
Displays the control registers which are specific to the 68020 and above processor. The value of these registers cannot be edited.

RF
Displays the registers which are specific to the FPU (optionally) present in 68020 and above processors. The value of these registers cannot be edited.
RM

Displays the registers which are specific to the MMU (optionally) present in 68020 and above processors. The value of these registers cannot be edited.

W (<register>)
Show/Edit the values in the CIA registers. This will display the register specified and allow the user to edit the current value. Pressing return will move to the next register. Use Escape to return to normal command mode. If no register is specified it starts at 0.

Y <addr>
This command is similar to the hex memory dump command and the ASCII memory dump command. This command displays the memory as binary starting at the address specified. The user is able to modify the value which is written back to memory upon pressing return. Escape will return to command mode.

YY <addr>
Show/Edit memory as binary. Similar to the Y command but initially displays 8 lines.

YYY <addr>
Show/Edit memory as binary. Similar to the Y command but initially displays 16 lines.

YS <bytes>
This command sets the number of bits displayed on each line when using the commands to Show/Edit binary data (Y/YY/YYY).

? expression
This command is a general use calculator. Any expressions will be evaluated and the result displayed. Operations are executed in standard order of priority with multiplication and division taking priority.

6. Music/Sample Ripping Commands

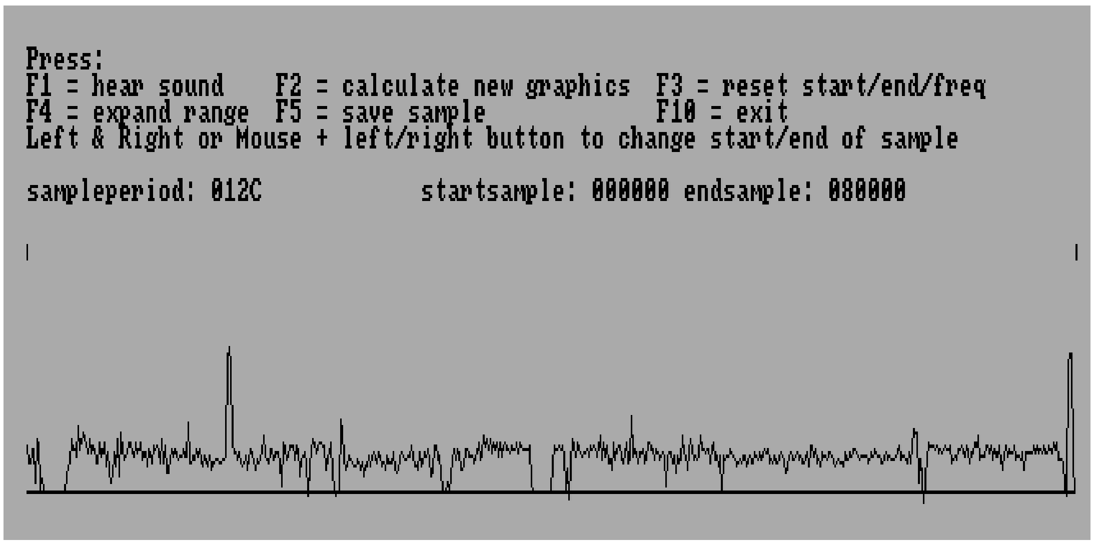
TRACKER (<start-addr>)
Searches for tracker modules in memory. If a module is found then the following keys can be used:

- F1 Play the module
- F2 Stop playing
- F3 See more details about the module
- F4 Save the module to disk
- F5 Rename the song
- F6 Show the song data
- F7 Continue searching for further modules
- F8 Change to old soundtracker 16 sample format
- F9 Calculate pattern length
- F10 Exit

SCAN
Display the chip memory as a waveform and allow the user to save the result to a file.
The following keys are used in the sample ripper:

- F1 Play the sound
- F2 Recalculate the waveform
- F3 Reset the start and end to the full chip ram
- F4 Expand the range
- F5 Save the sample
- Spacebar - switch between start and end markers
- Left Arrow - move the current marker down in memory
- Right Arrow - move the current marker up in memory
- Up Arrow - increase the period of the sample
- Down Arrow - decrease the period of the sample

You can also use the mouse to move the start and end markers while holding down the left or right mouse button.



```
Press:
F1 = hear sound    F2 = calculate new graphics  F3 = reset start/end/freq
F4 = expand range  F5 = save sample             F10 = exit
Left & Right or Mouse + left/right button to change start/end of sample

sampleperiod: 012C           startsample: 000000 endsample: 080000
```

Action Replay 5 Sample Ripper

7. Graphics Ripping Commands

P <picnr>
Show the current picture in mempeeker (the picture viewer). When viewing the current picture you can use a number of keyboard shortcuts to manipulate the screen to achieve the desired output. You can then save the picture using SPM.

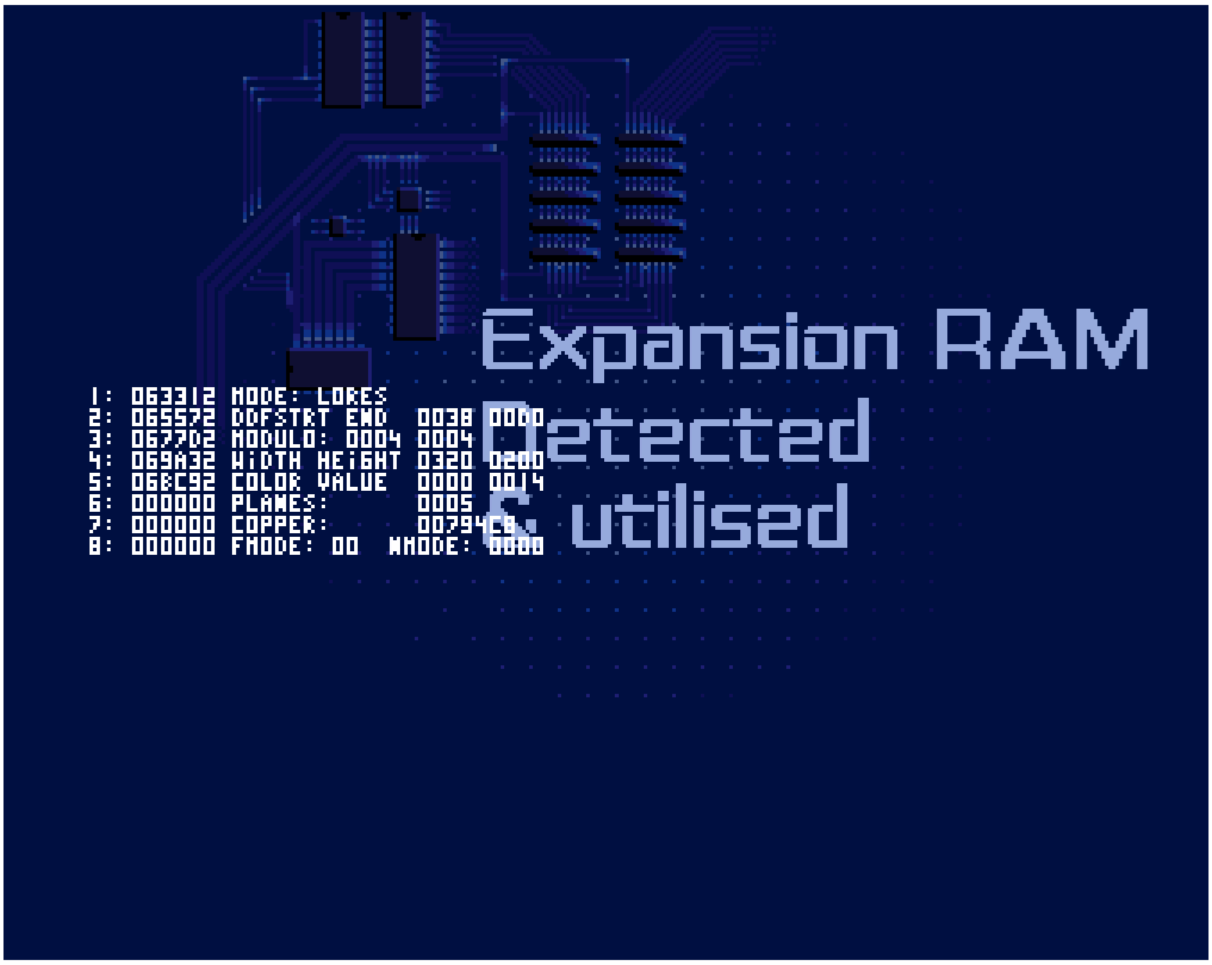
- a - Autoplane
- b - Brightness plus
- SHIFT b - Brightness minus
- c - Colorreg +1
- SHIFT c - Colorreg +16 (AGA)
- d - Dual playfield mode on
- SHIFT d - Dual playfield mode off
- e - Right border plus
- SHIFT e - Right border minus
- f - Fast plane up
- SHIFT f - Fast plane down
- g - Interlace mode on
- SHIFT g - Interlace mode off
- h - Hold and modify (ham) on
- SHIFT h - Hold and modify (ham) off
- i - Invert all colors
- l - Lores mode on
- SHIFT l - Hires mode on
- m - Modulo 1+2 plus
- n - Modulo 1+2 minus
- o - Modulo 1 minus
- SHIFT o - Modulo 2 minus
- p - Modulo 1 plus
- SHIFT p - Modulo 2 plus
- q - Clr modulo 1+2
- r - Rotate planepointer
- s - Left border minus
- SHIFT s - Left border plus
- w - White help screen
- SHIFT w - Black help screen
- x - Colorreg -1
- SHIFT x - Colorreg -16 (AGA)
- y - Switch diw and ddf mode
- 0 - Unlock all planes
- SHIFT 0 - Lock all planes
- 1 - Lock plane 1
- SHIFT 1 - Unlock plane 1
- 2 - Lock plane 2
- SHIFT 2 - Unlock plane 2
- 3 - Lock plane 3
- SHIFT 3 - Unlock plane 3
- 4 - Lock plane 4

- SHIFT 4 - Unlock plane 4
- 5 - Lock plane 5
- SHIFT 5 - Unlock plane 5
- 6 - Lock plane 6
- SHIFT 6 - Unlock plane 6
- 7 - Lock plane 7
- SHIFT 8 - Unlock plane 8
- + 1 bitplane plus
- - 1 bitplane minus
- = Set all bitplanes to bitplane1
- F1 - Default colors
- F2 - Random colors
- F3 - Reset
- F4 - Red color plus
- SHIFT F4 - Red color minus
- F5 - Blue color plus
- SHIFT F5 - Blue color minus
- F6 - Green color plus
- SHIFT F6 - Green color minus
- F10 - Set chosen picture into current program
- LEFT - Rotate picture left
- RIGHT - Rotate picture right
- UP - Scroll picture up
- SHIFT UP - Scroll picture up fast
- DOWN - Scroll picture down
- SHIFT DOWN - Scroll picture down fast
- DEL - Hide help screen
- ESC - Quit mempeeker
- HELP - Show help screen
- LEFT MOUSE BUTTON - Picture height plus
- RIGHT MOUSE BUTTON - Picture height minus

Move the mempeeker help screen with the mouse.

SP (<path>)<file>(,<nr> <height>)
Saves the current picture to disk.

SPM (<path>)<name>
Saves the current picture as previously displayed in the picture viewer (mempeeker).



```
1: 063312 MODE: LORES
2: 065572 DDFSTRT END  0038 0000
3: 0677D2 MODULO: 0004 0004
4: 069A32 WiDTH HEiGHT 0320 0200
5: 06BC92 COLOR VALUE  0000 0014
6: 000000 PLANES:      0005
7: 000000 COPPER:      00794C8
8: 000000 FMODE: 00  WMODE: 0000
```

Action Replay 5 Picture Viewer with help overlay displayed

8. Trainer Commands

TS <start-lives> (<start-address>)
Initiate the trainer search process. Starts searching for any addresses that contain the start-lives value. Optionally you can specify the start address of the search

T <lives>
Continue the trainer search process with an updated lives value. This will display the addresses where the search value matches and you can continue to use this command each time the lives value changes and the address list will be reduced.

TX
Exit the trainer search mode.

TF <address> (<start-address>) (<end-address>)
Search for any assembly language opcode that decrements the specified address. Use this after you have narrowed down the possible trainer addresses using the TS and T commands. It is also possible to specify a start and end address for the search.

TFD <address> (<start-address>) (<end-address>)
Similar to TF in that it searches for decrementing instructions but it also automatically removes the instruction. You may also optionally also provide a range for the search.

PC <picnr>
Displays the current frozen screen so that you can see how many lives or how much energy you currently have.

TD
Displays the complete list of deep trainer addresses that is being inspected.

TDC
Tells the deep trainer that the current amount of energy or lives has changed. The possible search address will be updated to reflect the changes.

TDD <start-addr> <end-addr>
Removes any deep trainer addresses within a range of memory locations specified.

TDI
List the current potential deep trainer address matches.

TDS
Start the deep trainer process.

TDX
Exit the deep trainer search mode.

Using the Trainer Commands

The best way to show you how to use the trainer command is by example. In the following example we are using the remarkable, but difficult, Rick Dangerous game (not 2) by Firebird. If you have not got the game then follow the outline anyway, it is similar in many games.

First load up Rick and start your man running away from the ball and freeze the game before you lose a life. Then type the line

```
TS6
```

as we start Rick dangerous with 6 lives and the following should appear

```
FIRST TRAINPASS!
CHANGE THE COUNT VALUE NEXT TIME!
SEARCHED UP TO :0572A6
TRAINMODE ACTIVE!
```

Now continue with the X command to restart the game and lose a life (shouldn't be too hard), then as you start your next life freeze the game again and type the following

```
T5
```

As you have now got five lives the screen will show

```
POSSIBLE ADDRESSES:
044972
SEARCHED UP TO :080000
TRAINMODE ACTIVE:
```

We now only have one possible location for the lives counter so this should be it. If it displays more than one address you should lose another life and use T4 until you are left with one possibility. Now to give yourself infinite lives type the following.

```
TFD 44972
```

After a brief pause, all being well the computer should respond with the following lines

```
SUB FOUND AT :00045E3C
SUBS ELIMINATED!
```

This has now removed the SUB1 instruction which decreased your lives. Use the X instruction to restart the game and Hey presto millions of Ricks. You could have also used the Monitor command M to change the value at 44972 to say 0B (that's 11 decimal) for 11 lives.

There follows a few more examples of how to use this feature.

GOLDEN AXE

First you must start the game and after a brief start sequence when you are able to move your character around the screen you should press the freeze button. Even though you start the game with 3 lives enter the line

```
TS 2
```

Now get back to the game using the X command and lose a life. This is the standard technique to the trainer; you will always lose a life between the TS and every T command. When you are on your second life, press the freezer button again and enter the line.

```
T 1
```

The screen will show some possibilities for locations. You must continue till you only have one possibility. Restart the game again and lose a further life then freeze again. Now enter the line.

```
T0
```

You may notice that more than one possible location is being displayed and we have no further lives to lose, the solution is simple. Exit to the game using the X command and lose your last life. The game will end but all you need to do is restart it. On your first life freeze the game and type the line

```
T2
```

Notice we do not use the TS even though we have restarted the game as we are still looking for the same lives location. Continue this, losing a life and refreezing until all of memory is searched. There will be only one possible location. If there is none and the trainer has failed you should try the whole process again as you have made a mistake on one of the number of lives. The location displayed should be 005955. The TFD command will not work on this game (try it if you like) so we must resort to a different technique to increase the number of lives. Type the line

```
M 5955
```

The screen will display the current number of lives (only valid if you froze the game while it was being played) as follows

```
005955 02 03 0c
```

with a whole lot of other numbers. The only important one is the 02, i.e. the first one after the location number. Use the cursor keys to move up to the 02 and change it to 7F (127 decimal) and press the return key when you have changed it. Press X to exit and you should have 127 lives which should be ample. If you come back to this game at a later date and wish to enter the cheat, you need not repeat the whole process again, simply start from the M command above.

RICK DANGEROUS 2

Start the game and as soon as Rick appears on the first screen and is able to move, press the freeze button and type the line

TS 6

Repeat the procedure X then lose another life and enter

T5

One further line should do it

T4

the screen will display only one possible address 0178AF. If you use the command TFD 0178AF nothing will be found but using a value one less will, i.e.

TFD 178AE

notice how leading zeros are not important but trailing ones are, as is the same with normal decimal numbers. This technique of using a value one less than that found using the T commands is very useful when the value is odd (i.e. ends in 1,3,5,7,9,B,D or F) and replacing this last digit with the even value one less i.e. (0,2,4,6,8,A,C,E respectively).

STRIDER 2

First start the game so your man is on the screen and able to move around, your lives counter will display 4, and freeze the game. In this case you should start the game with a value one more than your life counter i.e.

TS 5

now exit the trainer using the X and lose a life, then refreeze and type the following

T4

the screen will display only one possibility. Repeating the process numerous times will prove that this is definitely the one you are after as the only location displayed will be 6AD5 - this is the correct one! TFD 6AD5 will not work so enter the following

TFD 6AD4

Action Replay will then remove the instruction involved in decreasing your lives. Now simply exit using X and you have infinite lives.

MIDNIGHT RESISTANCE

After reading through the previous examples you should be ready for one or two abbreviations. Between each of the following T instructions you should lose a life and then refreeze. If you lose so many lives that the game is over simply restart the game and carry on. Do not use the TS command though.

```
Start game
TS 2
T 1
T 0
Restart game
T2
T1
T0
```

You will notice that however many times you lose a life there are always two possible locations. This is probably due to the way the programmer handles two players. The only way to find out which is the correct one is to try them. The two values are 11692 and 11767. Typing the following

```
TFD 11692
```

and testing using X and playing the game will have the desired effect so there is no need to try the other value.

ESWAT
This is abbreviated in the same way as above. After each TS and T command you should lose a life and re-freeze.

```
Start game
TS 2
T 1
T 0
Restart game
T 2
```

again the TFD command will not work so you must change the number of lives manually. Type

```
M 1BC57
```

the screen will look something like the following

```
001BC57 01 00 00 etc.
```

and a whole line of numbers. Changing the 01 to 7F (it does not have to be 01 but it is the FIRST number on the line after the address) will give you 127 lives. Remember to press return after you have changed the line.

NITRO
This is a good example of how the trainer can be used to find items that are not as predictable as lives and can go up as well as down. This cheat is for money that you use for buying cars, fuel etc. First you must finish the first round and gain some money on your way. When you get to the parts shop, the screen will show you how much money you have got. Now freeze the game. If you have 6 units of money, say, start the trainer with

```
TS 6
```

If you had 8 units of money you would of course use TS 8 etc. Restart the game, buy a couple of items to change the remaining money. When I tried it it went down to 3. Again refreeze the game and type the line

```
T 3
```

or whatever. Repeating this procedure a couple of times should give you a single value, if not you should do another lap and get some more money and keep trying till you get the value 1FBC7. There is a difference to your normal infinite lives in that you do not really want to stop decrementing money (although it would help if you have no money to start with you cannot buy anything). The best way is to use the M command as follows

```
M 1FBC7
```

then change the first number after the 1FBC7 to a value of say 50 for 50 units of money. Remember to press enter after you have changed the line.

NIGHTBREED (arcade action)
To show you that once you have found the infinite lives location in a game you need not search for it again type the following

```
TFD 24A
```

and you will get infinite lives. The TS, T commands can be skipped out in all the previous examples if you like, the whole procedure is there for example only. This location was found using TS3, T2, T1, restart, T3, T2 and subtracting one from the number found.

ROBOCOP 2
Use the following in the normal way:

Start game
TS 2
T 1
T 0
Restart game
T 2
TFD 8034 (one less than the value found)

JAMES POND
TS 3
T 2
T 1
TFD 1B0 (1B1 does not work)

BATMAN THE MOVIE
This was found using the standard technique, type the following
TFD 7C876

YOGI'S GREAT ESCAPE
This was found using a slightly different technique involving a little knowledge of machine code. I will only mention that the FA command was used, a very useful command for the hacker. To get infinite lives use the monitor command M 7B5E6 and alter the first 6 numbers as follows

```
7B5E6 4E 71 4E 71 4E 71
```

then press return and you will have infinite lives.

9. Program Freezing Commands

SA (<path>)<name>,<crate>)
Save the frozen program to disk. The memory, processor and hardware register states are compressed and saved to a file. The crate parameter is the compression rate and the valid range is between 0 and !65535.

SR (<path>)<name>,<crate>)
Saves the current state and then restarts execution. The parameters are the same as the SA command.

LA (<path>)<file>
Loads a file previously created using the SA or SR commands. The saved state is restored ready to restart.

LR (<path>)<file>
This instruction loads a save state file and resumes execution.

SLOADER
Saves a copy of the ALOAD loader program which can be used to load a saved state file without needing to use the Action Replay (or even to have Action Replay present).

LQ
Loads saved state from RAM. If the current program state has been saved to RAM (using SQ or SQR) then this command will restore the saved state.

LQR
Similar to LQ command but immediately resumes execution after the state is restored.

SQ
Save the current state to RAM. This requires an area of memory defined where the state can be saved. If you use the memory manager to disable an area of RAM then this can be used to save the state into, alternatively use SQMEM to define where the state is saved.

SQR
Save the current state to RAM and then resume execution. Similar to SQ and the same memory limitations apply.

EXQ
Effectively a combination of SQ and LQ, swaps the saved state in RAM with the currently running program.

EXQR
As per EXQ but immediately restart after the exchange.

SQMEM (<0/start>)
Specifies the address where state will be saved when using SQ, SQR, EXQ commands. Use 0 to return to the default where no address is defined.

10. System Information Commands

INTERRUPTS
Display the list of interrupts defined in the Exebase structure.

EXCEPTIONS
Shows a list of the 68000 exception vectors and their current addresses.

EXECBASE
Shows the complete Execbase structure in memory.

AVAIL
Displays the currently available system memory.

INFO <picnr>
Displays various information regarding the current system configuration and state.

LIBRARIES
Shows the library list from the Execbase structure.

RESOURCES
Shows the resource list from the Execbase structure.

CHIPREGS
Shows a list of the hardware registers which includes a name and offset for each register.

DCHIP <registername>
Displays detailed information about individual hardware registers.

DEVICES
Shows the devices list from the Execbase structure.

TASKS
Shows the tasks list from the Execbase structure.

PORTS
Shows the ports list from the Execbase structure.

11. Misc Commands

RAMTEST <start-addr> <end-addr>
Performs a two pass RAM test on the range of memory specified.

PACK <start-addr> <end-addr> <dest-addr> <crate>
Packs an area of memory denoted by the start and end addresses and saves it to the destination. The compression rate is controlled by the crate parameter and should be between 0 and !65535

UNPACK <dest-addr> <end-of-packed-addr>
Unpack an area of memory that was packed using the PACK command.

COLOR (<back-color> <pen-color>)
Sets the screen background and pen colours. This can also be done via the F3 preferences menu. If the command is run without any parameters the current settings will be displayed.

RCOLOR
Reset the pen and background colours to the original blue on white used by Action Relay Mk2 and 3.

RESETCFG
Resets all preferences back to the default values.

TM
Show the currently defined remarks. Remarks allow you to make notes about specific addresses related to a running program. Remarks are also saved as part of the save state files.

TMS <address>
Set a remark about an address. These can be useful as a reminder of an address used as a lives counter for example. There are 10 slots available for remarks.

TMD <address>
Delete a remark on a specific address.

SPR <nr|addr> (<nr|addr>)
Show and Edit sprite data. The sprite data is shown as text and can be edited in a similar way to the memory editors (updating the values and presing return to save them). The displayed numbers represent the sprite pixel colour values and may be from 0 to 3.

VERSION
Show current Action Replay firmware version information.

MEGASTICK (1)
Invokes the megastick configuration menu. Sets the joystick up to act as certain keys. Once you have finished, press Escape to exit the menu. Use the parameter option 1 if you only wish to select player 1.

NOSTICK
Disables the megastick joystick handler.

CLRSTICK
Clear the megastick values to 0.

LSTICK (<path>)<file>
Load the megastick values from a previously saved file.

SSTICK (<path>)<file>
Save the current megastick values to a file.

RESET
Reset the Amiga. Similar to pressing CTRL-Amiga-Amiga.

PAL
Switch the display to PAL mode. The display mode will be left as PAL when you exit Action Replay (unless the software you are running changes it back - for example as part of the copper list).

NTSC
Switch the display to NTSC mode. The display mode will be left as NTSC when you exit Action Replay (unless the software you are running changes it back - for example as part of the copper list).

SETMAP
Brings up the keymap editor. You can redefine the keyboard layout and save it to a file. You can also install the keymap so that when you exit Action Replay the new keymap will be activated. This function only works on Kickstart versions 1.x

KEYMAP US|UK|DE|IT
Sets the keymap used within the action replay console. Similar to F9 which cycles through the possible keymaps but with this command you can select the one you want directly.

ASCII
Display the ASCII character set. Useful if you need to look up the ASCII code for a particular character.

ALERT <guru-number>
Displays information regarding the guru alert codes (or software failure on later Kickstart versions). The Alert command with no guru number will display a list of the known codes and descriptions.

DISKIO <addr>
Install a copy of the Rob Northen DiskIO routines into memory. A brief description of how the routines can be used will also be displayed.

DOSIO <addr>
Install a copy of the Rob Northen DosIO routines into memory. A brief description of how the routines can be used will also be displayed.

ARRAM
Display the amount of memory that is installed in the current hardware device. The original Action Replay MK3 will have 40kb of RAM. Modified hardware versions may display 64K (WinUAE emulation allocates 64K of memory). The DeMoN cart will display 1024K. The Action Replay firmware is able to make use of however much memory is available to enhance the user experience.

FLASH (<path>)<file> (DeMoN hardware only)
Loads an updated firmware file into memory and flashes the file to the flash memory chip on the device. After the flash has completed you will need to reset your computer. If the flash fails or power is removed during flashing you may end up with a corrupted firmware. If this happens you may need to use the standalone firmware flashing program to rectify the situation.

SAVECFG (DeMoN hardware only)
Saves the current preferences to the flash memory of the cartridge. At power up the preferences will automatically be restored.

SYSRAM
At bootup the Amiga OS performs an extensive scan for additional memory. Each Kickstart version (and sometimes specific version of Kickstart for a particular machine) performs a slightly different scan. Eg the A4000 will scan for memory on the motherboard. This command displays the memory areas that have been detected by the operating system. Action Replay performs a similar scan but it is limited to a few commonly used address ranges (Chip ram, slow ram and auto-config fast ram at $200000).

SY <start-addr> <end-addr>
This command utilises the serial port of the amiga and sends a block of memory using the YMODEM protocol. You should use a fully wired serial cable capable of RTS/CTS handshaking.

SFY (<path>)<file>
This is similar to the SY command but rather than sending a block of memory it sends the contents of a file directly.

RY <start-addr>
This command uses the serial port to receive data using the YMODEM protocol. Each file received will be stored sequentially in memory starting at the address specified.

RFY (<path>)
Receive files via the serial port and save them directly to a file. Each file received will be saved to the path specified. If no path is specified then the files will be saved to the current path.

SERSPEED (<baud>)
Sets the baud rate for serial communications. This will affect the SY, SFY, RY, RFY and SER commands. The maximum baud rate that can be handled will be limited by the CPU in most cases. It is not recommended to go above 38400 on a standard 68000 based Amiga. If no baud rate is specified the current setting will be displayed.

SER
Toggle the serial console IO. When the serial console is enabled the console output is echoed to the serial port and input from the serial port will be treated as keypresses. Some screens cannot be displayed when the serial console is active (eg the preferences screen).

When connecting to the Amiga over serial using a terminal client it is possible that the screen sizes will not match and issues will occur when scrolling and with cursor movement. The two systems can get out of sync. The best solution to this happening is to use F1 on the Amiga side to clear the screen.

CRC16 <start-addr> <end-addr>
Calculates a 16 bit checksum value of the specified memory region. Useful for checking if an area of memory has changed.

CRC32 <start-addr> <end-addr>
Calculates a 32 bit checksum value of the specified memory region. Useful for checking if an area of memory has changed.

AXFER
Exits the action replay console and jumps into the ROMWACK or SAD system debugger (this debugger is present in every Kickstart ROM but whether you have ROMWACK or SAD will depend on your Kickstart version). This is useful if you wish to use the AmigaXfer software and have a suitable serial cable to connect your Amiga to a PC.

LED
Toggles the LED / Audio filter.

KILLMEM
Kills the currently running program and allocates a large track buffer using the memory. This is useful if a command you are running complains about not being able to allocate a buffer but doesn't give you the option to kill the running program.

IMODE <0|1|2|3> (Software Action Replay Only)
This command controls the triggering mechanism for the Action Replay software.

0 - Left & Right Mouse Button + 'F' Key
1 - '*' Key on Keypad
2 - Right Mouse Button
3 - Level 7 Int./button

KILL (Software Action Replay Only)
Perform a reset. If you are using the software based Action Replay then this will remove the system from memory.

ALLEXC (Software Action Replay Only)
Toggles the ability of the Action Replay being triggered.

DEEPMW
Toggles the Deep Memory watch functionality.

ROMAVOID (Software Action Replay Only)
Toggle that prevents the Action Replay device from being triggered when executing code from the Kickstart ROM area.

KICKROMADR (Software Action Replay Only)
Sets the address that Action Replay considers to be the Kickstart ROM area. Used with the ROMAVOID command.

CACHE (newval)
Enable or disable the CPU caches. Running the command with no parameter displays the current state of the cache. Does not work unless your CPU has caches. The value is written directly to the CACR register. Refer to the 680x0 manual for your processor for further information.

NORMALCHAR
Normal printerchars. Requires an EPSON compatible printer.

SMALLCHAR
Activate very small printer chars. Requires an EPSON compatible printer.

PRT <string>
Prints a string to the attached printer. Hex codes may also be used to send specific control characters to the printer.

VIRUS
Search memory for any known virus. If the virus checker is enabled in the preferences then this command is not needed as you will always be warned upon entering the Action Replay console.

KILLVIRUS
Search and remove any known virus from memory.

12. The Standalone Flasher Tool

This utility is for flashing the DeMoN cartridge. It relies on the presence of the AT29C010A flash chips which are used to store the firmware on this device.

While the cartridge can be flashed from the Action Replay console using the FLASH command it may be necessary to flash the firmware without using the device itself. The DeMoN cart has an anti-brick mechanism which allows it to be flashed even if the firmware has become corrupted and is not functional.

There are two jumpers on the PCB that allow the device to be flashed under these circumstances.

The first jumper (labelled JP1) which is set to 1-2 under normal operation enables the stealth capabilities of the cart. In this mode the ROM and RAM present on the cart is not visible to the Amiga system until the cart is activated. This means software is not able to detect the presence of the cart. In order to flash a new firmware to the cart this jumper should be set to 2-3 and this will make the DeMoN memory visible at all times.

IF you are able to boot your machine successfully then with this JP1 jumper correctly set you can use the standalone flasher to reprogram the firmware of the cart. If your machine does not boot with the DeMoN cart plugged in then the firmware is most likely corrupted and is not initialising correctly. In this scenario JP2 comes into play. Setting JP2 to 2-3 will deactivate the cart functionality but the memory will still be visible. This means the system can be booted and then the DeMoN can be flashed. Once the flashing is complete the JP2 must be returned to 1-2 for normal operation. It is also recommended to return JP1 to its standard setting of 1-2 as well but this is not mandatory, the cart will function with either setting of JP1 but the stealth features will not be operational.

In summary the two jumpers on the board function as follows:

```
JP1

      1 - 2  - 3
      |        |
      Stealth Visible


JP2
      1 - 2  - 3
      |        |
      Enabled          Disabled
```

both jumpers should be set to 1-2 for normal operation.

To run the standalone flasher utility you should create a floppy disk with just the standalone command line flasher utility (ar5flasher) and a copy of the firmware file you wish to flash.

Install the standard bootblock on the disk but do not create a startup-sequence. Booting from this floppy will then result in the user being dropped to the CLI prompt. Simply run the flasher tool from this CLI prompt passing in the name of the ROM file as a parameter. This will look something like this (exact text may vary according to the version of the Amiga OS being used):

```
Copyright @1987 Commodore-Amiga, Inc.
All rights reserved.
Release 1.3
1> ar5flasher ar5firmware.rom
```

After entering this and pressing return the flasher should perform its job and the ROM file will be applied to the cart flash memory. A verify operation will also take place to ensure it was completed successfully

If the operation completes successfully you may return the cart jumpers to your desired setting and continue to use the DeMoN cart as desired.

If the flash is not successful you will be given an appropriate error message to indicate where the issue lies.

If the device is functioning correctly this process should allow a reflash of the firmware and recovery from a brick where there firmware has become corrupted.

13. The Built in API commands

One of the new features of Action Replay 5 is an API that allows some of the functionality of the Action Replay firmware to be called from code. This can be useful if you wish to display debug information to the Action Replay Console or if you wish to make use of the disk routines built into the Action Replay for example.

The API is activated using the SETAPI command and then the api can be called using the TRAP #7 instruction from your own code. The function number should be passed in D0.W

Function Number	Description	Parameters
-1	End API Stop executing code and return to the Action Replay command prompt.	None
0	Print Text The text is printed to the current output page of the action replay console.	A0 = Text to print (null terminated)
1	Print Value The hex representation of the value in D1 is printed. The least significant digits up to the maximum specified are output.	D1 = Value to print D2 = Number of digits to print
2	Clear Screen Clear the currently selected output page.	None
3	Select output page Select the specified output page.	D1 = output page (1/2)

4	Load a file
	
	Load a file to the specified address. The filename should be formatted in the usual action replay format.
		A0 = filename
		A1 = load address

Function Number	Description	Parameters

5	Save a file
	
	Save a file from the start address to end address-1. The filename should be formatted in the usual action replay format.
		A0 = filename
		A1 = start address
		A2 = end address

6	Save a data file
	
	Save a file in the format of DC.B statements from the start address to end address-1. The filename should be formatted in the usual action replay format.
		A0 = filename
		A1 = start address
		A2 = end address

7	Read tracks
	
	Read tracks from the current drive into memory.
		D1 = start track
		D2 = track count
		A0 = load address

8	Write tracks
	
	Write tracks from memory to the current drive in standard AmigaDOS format.
		D1 = start track
		D2 = track count
		A0 = save address

9	Read Pdos tracks
	
	Read pdos formatted tracks from the current drive into memory.
		D1 = start track
		D2 = track count
		D3 = pdos key (0 to calculate automatically)
		A0 = load address

10	Write Pdos tracks
	
	Write pdos formatted tracks from memory to the current drive.
		D1 = start track
		D2 = track count
		D3 = pdos key
		A0 = save address

11	Read sectors
	
	Read sectors from the current drive into memory.
		D1 = start sector
		D2 = sector count
		A0 = load address

Function Number	Description	Parameters

12	Read Pdos sectors
	
	Read pdos formatted sectors from the current drive into memory.
		D1 = start sector
		D2 = sector count
		D3 = pdos key (0 to calculate automatically)
		A0 = load address

13	Read bytes
	
	Read any number of bytes from the current drive into memory.
		D1 = start byte offset
		D2 = byte count
		A0 = load address

14	Read Pdos bytes
	
	Read any number of bytes from the current drive into memory from a pdos format disk.
		D1 = start byte offset
		D2 = byte count
		D3 = pdos key (0 to calculate automatically)
		A0 = load address

15	Read Mfm track
	
	Read data in raw mfm format from the current drive into memory.
	
	Note that the initial sync marker is written to memory to allow it to be written back to disk. This means the number of words transferred will need to be 1 higher than the value written to DSKLEN
		D1 = start track
		D2 = track count
		D3 = sync marker
		D4 = track length (in words)
		A0 = load address

16	Write Mfm track
	
	Write raw mfm data from memory to the current drive.
		D1 = start track
		D2 = track count
		D3 = track length (in words)
		A0 = save address

14. Version Hsistory

Firmware
v5.0.0 - Initial release (05-Apr-2025)

Ar5Flasher
v1.0.0 - Initial release (04-Apr-2025)

Ar5Loader
v1.0.0 - Initial release (05-Apr-2025)