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 has many new features and bug fixes and can run in a number of different configurations and will provide slightly different capabilities depending on the system it is running on.

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.

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).

Software Debugger 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.

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.

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 lisst 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

(cont)
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.

Memory Control (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.

Module Interna

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.

Megastick
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.

Autoconfig
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.

Burst Nibler
The Burst Nibbler faststart option allows entry to the nibbler screen 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.

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).

ED (<path>)<file>
Opens the file in the built in text editor.

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.

DISKWIPE <drive>
Performs a quick wipe of the specified 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>
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.

BOOTPROT <codenumber)
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>
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 wll become unreadable so take care when using this command.

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.

BURST
Launch the burst nibbler tool. 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 burst nibbler. You must reset once you are done with the tool. Burst nibbler is a fully featured disk copy program that is capable of copying non-dos disks. The detailed usage of this tool is outside of the scope of this manual.

CODECOPY <source-drive> <dest-drive>
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 trackbuffer 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> (<sectornum-offset>)
MFM (Decode raw mfm data)

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
SETEXCEPT (Set exception handler)

SETAPI
SETAPI (Set api handler)

CLRAPI
CLRAPI (Remove API handler)

COMP <start-addr> <end-addr> <dest-addr>
COMP (Compare memory)

LM (<path>)<file>,<dest-addr>
LM (Load file to memory)

SM (<path>)<name>,<start-addr> <end-addr>
SM (Save memoryblock to disk)

SMDATA (<path>)<file>,<start-addr> <end-addr>
SMDATA (Save memoryblock to disk as data)

SMDC (<path>)<file>,<start-addr> <end-addr>
SMDC (Save memoryblock to disk as dc.b)

A <address>
A (Assemble)

B
B (Show breakpoints)

BS <addr>
BS (Set breakpoint)

BD <addr>
BD (Delete breakpoint)

BDA
BDA (Delete all breakpoints)

MS <address>
MS (Set memwatchpoint)

MW
MW (Display memwatchpoints)

MD <address>
MD (Delete memwatchpoint)

MDA
MDA (Delete all memwatchpoints)

ST <steps>
ST (Trace current program (also subs))

TR <steps>
TR (Trace current program (not subs))

X
X (Restart current program)

C <1|2|address>
C (Copper disassembler)

DBG
DBG (Launch debugger)

D (<addr>)
D (Disassemble)

DD (<addr>)
DD (Disassemble 8 lines)

DDD (<addr>)
DDD (Disassemble 16 lines)

E (<offset>)
E (Show/edit chip registers)

EA
EA (Show full AGA pallete)

F <string>(,<start-addr> <end-addr>)
F (Search For string)

FS <string>(,<start-addr> <end-addr>)
FS (Search for string case insensitive)

FA <addr> (<start-addr> <end-addr>)
FA (Find addressing opcode)

FAQ <addr> (<start-addr> <end-addr>)
FAQ (Find addressing opcode quick)

FR <string>(,<start-addr> <end-addr>)
FR (Search for relative-string)

FC (<start-addr <end-addr>)
FC (Search for Rob Northen Copylock)

CI <addr>
CI (Display copylock info)

G (<addr>)
G (Restart program at address)

GK (<addr>)
GK (Disable DMA/Interrupts and restart program at address)

TRANS <start-addr> <end-addr> <dest-addr>
TRANS (Copy memoryblock)

WS <string>, <start-addr>
WS (Write string to memory)

M <address>
M (Show/edit memory as bytes)

MM <address>
MM (Show/edit memory bytes - 8 lines)

MMM <address>
MMM (Show/edit memory bytes - 16 lines)

MEMCODE <start-addr> <end-addr> <code>
MEMCODE (EOR.B encode memory)

ADD <start-address> <end-address> <value>
ADD (Adds a value to memory range)

N <address>
N (Show/edit memory as ASCII)

NN <address>
NN (Show/edit memory as ASCII - 8 lines)

NNN <address>
NNN (Show/edit memory as ASCII - 16 lines)

NO <offset>
NO (Show/set ascii-dump offset)

MQ <address>
MQ (Display memory quick as Hex/ASCII)

NQ <address>
NQ (Display memory quick as ASCII)

O <string>, <start-addr> <end-addr>
O (Fill memoryblock with string)

ROBD
ROBD (Enable/disable Rob Northen decryptor)

R (<reg> <value>)
R (Show/edit processor registers)

RC
RC (Show 020+ control registers)

RF
RF (Show FPU registers)

RM
RM (Show MMU registers)

W <register>
W (Show/edit CIA data)

Y <addr>
Y (Show/edit memory as binary)

YY <addr>
YY (Show/edit memory as binary - 8 lines)

YYY <addr>
YYY (Show/edit memory as binary - 16 lines)

YS <bytes>
YS (Show/set datawidth for the Y command)

? expression
? (Calculator)

6. Music/Sample Ripping Commands

TRACKER (<start-addr>)
TRACKER (Rips soundtracker-modules in memory)

SCAN
SCAN (Scan memory for samples)

7. Graphics Ripping Commands

SP (<path>)<file>(,<nr> <height>)
SP (Save current picture to disk)

SPM (<path>)<name>
SPM (Save picture of memory-peeker)

P <picnr>
P (Show current picture/mempeeker)

8. Trainer Commands

TS <start-lives> <start-address>
TS (Start trainer/trainermode)

T <lives>
T (Show addresses/continue trainer)

TX
TX (Exit trainermode)

TF <address>
TF (Search for decrementing opcodes)

TFD <address>
TFD (Search and remove decrement opcodes)

PC <picnr>
PC (Show current picture + energy count)

TD
TD (Display deep trainer addresses)

TDC
TDC (Deep trainer change count)

TDD <start-addr> <end-addr>
TDD (Deep trainer delete addresses)

TDI
TDI (Display probable trainer addresses)

TDS
TDS (Deep trainer start count)

TDX
TDX (Exit old deep trainer)

TM
TM (Show remarks about curr. program)

9. Program Freezing Commands

SA (<path>)<name>,<crate>)
SA (Save current program to disk)

SR (<path>)<name>,<crate>)
SR (Save current program and start)

LA (<path>)<file>
LA (Load freezefile from disk)

LR (<path>)<file>
LR (Load freezefile from disk and start)

SLOADER
SLOADER (Save loader to active drive)

LQ
LQ (Load all from ramdisk)

LQR
LQR (Load all from ramdisk and restart)

SQ
SQ (Save all to ramdisk)

SQR
SQR (Save all to ramdisk and restart)

EXQ
EXQ (Exchange prg with ramdisk prg)

EXQR

EXQR (Exchange prg with ramdisk prg and run)

SQMEM (<0/start>)
SQMEM (En/disable savequick in fastmemory)

10. System Information Commands

INTERRUPTS
INTERRUPTS (Show execbase interrupt-list)

EXCEPTIONS
EXCEPTIONS (Show exception and interrupt vectors)

EXECBASE
EXECBASE (Show execbase structure)

AVAIL
AVAIL (Display available memory)

INFO <picnr>
INFO (Show system parameters)

LIBRARIES
LIBRARIES (Show execbase library-list)

RESOURCES
RESOURCES (Show execbase resource-list)

CHIPREGS
CHIPREGS (Display chip register info)

DCHIP <registername>
DCHIP (Display register info)

DEVICES
DEVICES (Show execbase device list)

TASKS
TASKS (Show execbase task-lists)

PORTS
PORTS (Show execbase port-list)

11. Misc Commands

RAMTEST <start-addr> <end-addr>
RAMTEST (Checks memory for errors)

PACK <start-addr> <end-addr> <dest-addr> <rate>
PACK (Pack memory)

UNPACK <dest-addr> <end-of-packed-addr>
UNPACK (Unpack packed mem)

COLOR <back-color> <pen-color>
COLOR (Set screen colours)

RCOLOR
RCOLOR (Reset colors)

RESETCFG
RESETCFG (Reset all preferences)

TM
TM (Show remarks about current program)

TMS <address>
TMS (Set remark about curr. program addr.)

TMD <address>
TMD (Delete remark about program)

SPR <nr|addr> (<nr|addr>)
SPR (Show/edit sprites)

VERSION
VERSION (Show version info)

MEGASTICK (1)
MEGASTICK (Joystick handler)

NOSTICK
NOSTICK (Remove joystick-handler)

CLRSTICK
CLRSTICK (Clear megastick values)

LSTICK (<path>)<file>
LSTICK (Load joystick-handler data)

SSTICK (<path>)<file>

SSTICK (Save joystick-handler data)

RESET
RESET (Exit AR and reset Amiga)

PAL
PAL (Set PAL display mode)

NTSC
NTSC (Set NTSC display mode)

SETMAP
SETMAP (Keymap editor)

KEYMAP US|UK|DE|IT
KEYMAP (Set keymap)

SETCOP <copper-addr>
SETCOP (Specify copper for exit)

ASCII
ASCII (Display ASCII character set)

ALERT <guru-number>
ALERT (Display alert info)

DISKIO <addr>
DISKIO (Copy Rob Northen DISKIO package to RAM)

DOSIO <addr>
DOSIO (Copy Rob Northen DOSIO package to RAM)

ARRAM
ARRAM (Display Action Replay RAM info)

FLASH (<path>)<file>
FLASH (Update firmware)

SAVECFG
SAVECFG (Save current to flash)

SYSRAM
SYSRAM (Show system memory blocks)

SY <start-addr> <end-addr>
SY (Send memory via serial ymodem)

SFY (<path>)<file>
SFY (Send files via serial ymodem)

RY <start-addr>
RY (Receive files to memory via serial ymodem)

RFY <path>
RFY (Receive files via serial ymodem)

SERSPEED <baud>
SERSPEED (Set serial baud rate)

SER
SER (Enable/Disable serial console)

CRC16 <start-addr> <end-addr>
CRC16 (Calculate a CRC16 checksum)

CRC32 <start-addr> <end-addr>
CRC32 (Calculate a CRC32 checksum)

AXFER
AXFER (Setup system for AmigaXfer)

LED
LED (Toggle filter/LED status)

KILLMEM
KILLMEM (Kill running program and allocate largest track buffer)

IMODE <0|1|2|3>
IMODE (Set interrupt mode)

KILL
KILL (Remove AR from memory)

ALLEXC
ALLEXC (Enable/disable all exceptions)

DEEPMW
DEEPMW (Enable/disable deep memwatch)

ROMAVOID
ROMAVOID (Enable/disable triggering from ROM)

KICKROMADR

KICKROMADR (Toggle kickstart ROM address)

CACHE <0|1>
CACHE (Enable/disable cache)

NORMALCHAR
NORMALCHAR (Normal printerchars)

SMALLCHAR
SMALLCHAR (Activate very small printer chars)

PRT <string>
PRT (Print string)

VIRUS
VIRUS (Search for virus in memory)

KILLVIRUS
KILLVIRUS (Search and remove virus)

Index