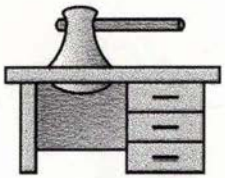
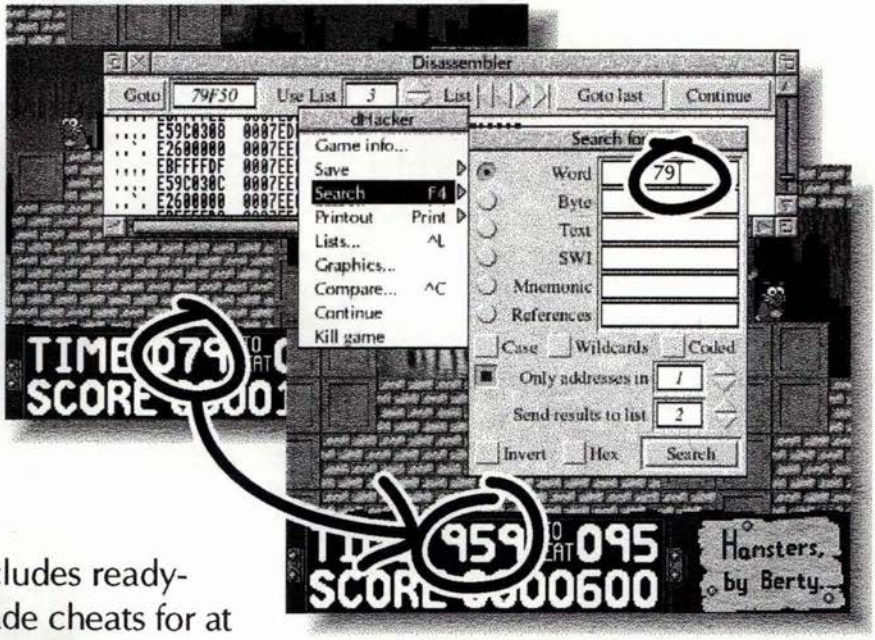


*desktop*Hacker

The last word in cheating for Acorn RISC computers



Includes ready-made cheats for at least 157 games!

DoggySoft

Software for Acorn RISC computers
Archimedes & Risc PC compatible
RISC OS 2 through 3.5 compatible
2 megabytes RAM recommended

desktop Hacker

user guide

About this guide

This is the first issue of the user guide, and refers to version 1.01 of Desktop Hacker. Any later versions will have a *ReadMe* file in the root directory of the disc, detailing any changes made.

Further, more specialised information can be gained from the *Documents* directory on the disc. For general explanation of the features, interactive help is available.

The *Textfiles* directory contains some textfiles giving hints, tips, and cheat modes for many Acorn games.

Introduction

Desktop Hacker's features generally fall into two categories. There are features intended for use inside the game, like slowdown and screenshot saving, and there are many number-crunching features like searching and comparing, to allow you to find cheats. A *cheat* is one or more alteration to a game's memory, producing the effect of making the game easier. For example, a game's lives counter could be altered to give a vast number of lives. Once you have found a cheat for a game, you can program it into a *cheat module*.

A cheat module is a small piece of code that runs without Desktop Hacker, and just executes the cheats programmed into it when certain keys are pressed. Because cheat modules are freely distributable, and they don't need a copy of Desktop Hacker to work, you can then give them to anyone you like, and they can use them. In fact, Desktop Hacker comes with a huge collection of ready-made cheat modules: look in the *CheatMods* directory on the disc. To use a CheatMod, double-click on it, and remember all the keys it tells you. If you forget, press F12 and type *CheatInfo*. Run the game, and activate the cheats by pressing the keys. To remove the CheatMod, press F12 and type *RMKill CheatMod*.

To hack a game, run Desktop Hacker, and instead of double-clicking on the game to run it, drag it to the Desktop Hacker icon on the icon bar. The game will run as usual and Desktop Hacker will be activated. You can now hold down both *Alt* keys at any time to interrupt the game. Doing so will bring you back to the RISC OS desktop, where it is possible not only to hack the game, but also to run all your other applications. Two windows will also appear: one shows a memory dump and *disassembly* of the game; another shows the contents of the game's registers, the screen mode it was using, and how much memory it was using. There is also a *button bar* at the top of the disassembly window, with some frequently used features. Other features are available from the main menu, which is obtained by clicking Menu on the disassembly window. When you have finished, you can return to the game by clicking Select on the Desktop Hacker icon bar icon, clicking on the *Continue* button on the button bar, or choosing *Continue* from the main menu.

Note that Desktop Hacker cannot work on 100% of games. Games authors will insist on using illegal methods; Desktop Hacker anticipates this as much as possible, but it's always possible that there will be something Desktop Hacker can't deal with. So always save your work before hacking a game. Games authors also will try to use as much memory as possible, so there may not be enough memory left on a one megabyte machine to run Desktop Hacker. We recommend having two megabytes or more to hack games.

Desktop Hacker disables the sound system during hacking, to prevent the game from attempting to produce sound from itself and crashing. You can turn the sound system back on by pressing F12 and typing *Audio On*, but this is very likely to crash the machine.

Games utilities

Some parts of games are just too tricky to play normally. Desktop Hacker can slow games down to allow you to get through these parts easily. Interrupt the game and open the icon bar menu: Move over the *Slowdown* item and choose a factor to slow the game down by — 2 would be half-speed, 4 quarter-speed, and so on. Select the option to turn slowdown on, and return to the game. Remember to turn slowdown off before loading any other games — slowdown slows loading down too!

Some games do not allow you to return to the desktop. If this is the case, you can interrupt them, open the main menu, and choose *Kill game*. Only use this as a last resort though; it'll leave the sound system silent, and not give the game a chance to tidy its modules up.

You may want to save a screenshot of a game, to prove you have reached a certain point, examine its graphics, or include it in a document of some sort. To do this, run the game until the screenshot appears that you wish to save. Interrupt the game, open the main menu, the *Save* submenu, and go over the *Screenshot* item. The screenshot can then be saved as a sprite in the normal way. Usually you will want to keep the *With palette* option on: this means that the colours used by the game will be saved with the sprite. However, some games do not use the normal RISC OS palette, instead programming the video controller themselves. In this case, the palette will be entirely black, so you won't be able to see anything. If this happens, save the sprite without a palette, and try to work out what the colours are supposed to be manually.

Many games do not have a save game option. This means that you have to start again from the beginning of the game, or, if the game has passwords, the beginning of the level, every time you die. Desktop Hacker can save an entire game position. This means that you can save the game just before a particularly tricky bit, and re-load if you fail, so you don't have to go through everything before it again. Also, saving a game means you can return to it at

a later date. To save a game position, interrupt the game at the point you wish to save, open the main menu, and the *Save* submenu. Go over the *Position* item, and drag the icon to save in the normal way. Note that the files saved here are very large: almost always larger than can be stored on an ADFS 800K disc. It is recommended that you save to a hard disc, or failing that, a high density ADFS 1600K disc, or a compressed filing system. To load a game back in, simply drag it to the Desktop Hacker icon. However, before loading a game file, try to get the game into as similar a state as possible to where it was saved; the closer, the more likely that it'll work properly. For example, if the position was saved in the middle of a game, do actually start the game before loading the position, it may not work from the title screen.

Many games store their graphics as raw blocks of data. Desktop Hacker allows you to find these blocks, and convert them into sprites, which you can look at and edit using *Paint*. Choose the *Graphics...* item from the main menu. A window will appear showing the keys to use, and having an option to use the palette. This option should normally be selected; games which don't use the RISC OS palette will need this off, as with screenshots. Click on *View* to start graphics ripping. Desktop Hacker takes over the whole screen and shows you the contents of the game's memory as graphics. It is up to you to find where the real graphics are stored; it is often easy to spot graphics data, as blocks of similar colours will stand out, and some sort of pattern is often visible. To move around the game's memory, use the arrow keys. Once you have found the start of some graphics data, hold down Shift and use the arrow keys to change the size of the graphics. Use Shift–Left first until the lines of graphics line up, showing a graphic from the game, then use Shift–Up to get its vertical height. You can speed up the process of changing the size or moving around by holding down Ctrl at the same time as the other keys. Once you are happy with the graphics on-screen, press Return, and drag the icon to save as normal. If there are more sprites after the first, all the same size (for example, a set of different letters), you can save them all in the same sprite file, by adjusting the *Sprites* field.

Many games have music stored in *SoundTracker*, *Tracker*, *Desktop Tracker*, and *Digital Symphony* formats. You can often rip them out and save them, using the *IdHMusRip* application. Use it when Desktop Hacker is hacking a game, and you can save out any music that it finds. The files can then be played using the many public domain music players.

The Choices window

This window allows you to configure certain parts of Desktop Hacker to your own taste. To get it, choose the *Choices...* item from the icon bar menu or click Adjust on the icon bar icon.

At the top of the window is the hot-keys field. The hot-keys are normally both *Alts*, but they can be changed to any combination. To do this, click on *Change* and hold down the keys required. This is essential for A4 portable users, who only have one Alt key.

To the left of the window, there's some disassembly options. If you have a preferred style of disassembly, you can choose it here. To the right are some options concerning Desktop Hacker's interface, which you may like to change. Click on *OK* to set the choices up, or *Cancel* to ignore any changes you made. *Save* saves them to disc to be used in the future, and *Default* resets them to the original values.

Essential technical information

In order to find cheats, you must know a little about the way Archimedes and Risc PC computers work. In particular, you must understand *hex*, and how memory is laid out.

Hexadecimal, or *hex* for short, is a way of counting often used in computing. Normally, we use a system called decimal. In this system, we count up to nine, and then ten is specified by going to two digits, 10. In hex, we count up to fifteen, and then sixteen is specified by 10. To prevent confusion, where a number is hex, it is normally preceded by an ampersand (&). The numbers ten to fifteen are shown by the letters *A* to *F*, A being ten and F fifteen. Hence &1C equals 16+12=28 in decimal.

The memory of a computer is split up into a large number of locations. Each location holds one *byte* of memory, and is referred to by a number, the *address*, which is *always* in hex, despite sometimes being referred to without the ampersand. The area of memory used by a game starts at address &8000, and ends at a point dependent on the amount of memory the game uses.

A byte of memory can store a number between zero and 255 (&FF). Obviously, 255 is not a big enough number for all purposes, so another form of memory can be used: the *word*. A word is made up of four bytes together, and it can store numbers between zero and about four billion, enough for most purposes. Since dealing with words is often easier than with bytes, Acorn games usually use them. One limitation you should bear in mind with words [illegible] ...vated by holding [illegible] on the editor [illegible] that activates words at [illegible] that the [illegible] many [illegible]

Desktop Hacker can search through [illegible] different things. When doing [illegible] addresses where these things were [illegible] search is finished, a window showing [illegible] automatically. This window is also available [illegible] menu and the icon bar menu.

To start a search [illegible] search [illegible] Or, [illegible] and are [illegible] to [illegible] you type in the word [illegible] hex rather than decimal. If you select the *Only* [illegible] in option, the resulting list will only contain addresses that fitted your criteria for the search and were in the [illegible] you chose here. Finally, click on *Search* to perform the search.

Word searches check every word of memory to see if it contains the value you chose. Similarly, *byte* searches check every byte of memory for the value. Searches for *references* disassemble every word of memory, and work

out what address the instruction stored there accesses, if any, comparing this to the supplied address. *SWI* searches check through memory for calls to any SWI routine (specified by name or number), or any group of SWIs (specified by putting the chunk name and underline, whthout a name afterwards), and are only of use if you have knowledge of SWIs. *Mnemonic* searches go through memory, disassemble each word into an instruction, as shown on the right hand side of the disassembler, and then checks to see whether the text you typed was to be found anywhere in the disassembled instruction. Any spaces are ignored. Hence, if you typed *UBCSR*, the instruction *SUBCS R3,R3,#1* would be matched. If the *Case* option is *not* selected, the case of the letters is not important; you can type a lower case search string, and upper case (capital) letters will still match. You can also include *wildcards* in the search string, if you select the *Wildcards* option. The wildcards in Desktop Hacker are: *$* which matches any character; *%* which matches any letter; *#* which matches any number and *&* which matches values 10 and 13, both newline characters. *Text* searches may be used to find addresses where many bytes together store some text; useful for finding passwords. The *Case* and *Wildcards* options both also work on text searches, and there is another option, *Coded*, which allows text strings encoded by a simple single-byte exlusive-OR to be found.

All hacking revolves around the process of guessing how the game works, and fiddling with bits of it to make it do things the programmer didn't intend. The simplest method of hacking is to find the address where a simple value like the number of lives you have is stored, and increase its contents, increasing your number of lives for example. This method will work for any countable value: lives, score, money, and so on.

To alter the contents of an address, click on it with Select in the disassembly window. A small editing window appear. You can edit the value in here to a different one, then click on *OK* to alter the memory. The value is always hex, and you can only edit a whole word at a time. To edit a byte, find the right two digits in the word and alter them. Which two digits are used depends on the address you wish to alter. If it's the same as the address shown by the disassembler, it is the rightmost pair. If the address is one above the disassembler's, it's the second pair from the right, and so on.

Clicking Adjust on the disassembler window toggles whether the address clicked on is in the selected list or not; similarly, clicking Adjust on an address in the lists window removes that address from the list. If you are disassembling a program, and come across an instruction to refers to another address, you can hold Shift and click Select to move the disassembler to the address it pointed to. When you have finished looking at that, you can return by clicking on Goto last.

When you have found the address for a counter, you can attempt to find the code in the game that affects that address. If, for example, you could find the code that decreased the lives counter, remove it, and obtain infinite lives. To search for code that affects the contents of an address, type the addresses into the *References* field and search for it. This will find any instruction that refers to the address you typed, even if it is referred to using an offset to a register. Step through the resulting addresses, and try to find a bit of code that has a LDR from the address, a SUB instruction, and a STR to the address close to each other. This will be the bit that decreases the counter — you may find more than one bit. The easiest way of stopping it from decreasing the counter is to replace the number in the SUB instruction with 0. To do this, click Select on the address to alter, and replace the rightmost pair of numbers with 0.

bu must know at least one password already. Type this into the *text* field and search for it. If you find any addresses, move to them. If you are lucky, a the rest of the passwords will be listed down the left hand side of the disassembler. However, many authors now encode their passwords, so you won't be able to see them easily. Desktop Hacker can deal with the simplest method of encoding, a single-byte EOR. To enable this, select the *Coded* option on the search window. When you move to the address, you won't be able to see all the passwords, as they are encoded. To decode them, you'll have to work out the encoding key, and decode each byte yourself, in BASIC.

Memory Compare

The *Memory Compare* window allows you to find all the addresses that have changed over a certain period of time. This is particularly useful when the commodity you are looking to find does not appear on the screen as an easily countable number. For example, an energy bar that runs down, or the amount of time you have left to be invulnerable. To use memory compare, you must first save out the game's application memory, from the *Save* submenu. This file serves as a reference to check against to see what has changed later, and it's often very large. Return to the game, change whatever it is you are searching for, and interrupt again immediately. The shorter the time between the interrupts, the less will have changed you are not searching for, and the shorter the lists you will have to deal with. Choose *Compare...* from the main

menu, and drag the application file you saved earlier into the Memory Compare window. Choose which list to send the addresses to, and whether to use bytes or words. Generally, you will want to have the *Signed* option set for words, but not for bytes; it specifies whether numbers are allowed to go negative. Choose the *Changed* option if you want to search for all addresses that have changed; if you are sure that the value you are searching for will have increased or decreased, you can select those options instead (for example, had you lost energy, the value would certainly have gone down).

You will inevitably get a very big list from memory compare, which needs to be narrowed down. You can do this by repeating the memory compare process more than once, generating many lists, and then using the list handling features to find suitable addresses. As well as generating lists of addresses that change when what you are looking for changes, you can generate lists of addresses that change even when what you are looking for stays the same, and eliminate them. Between interrupts, do as much as possible, but make sure that what you are looking for is not changed. Then look for all addresses that have changed, and that will eliminate a lot of addresses which change constantly.

List handling

The list handling features become particularly useful when using memory compare, to cross-reference its results, to give one list, as short as possible, that gives all possible addresses for what you are looking for.

Open the lists window from the main menu or icon bar menu. At the top, you can add single addresses to the selected list, and remove all addresses at once. At the bottom, you can add or remove blocks of addresses, which can be useful if you are sure an address you are looking for lies within, or outside of, a certain range. In the middle lie the list processing features. These are what you use to cross-reference lists.

All processes need a second list to be specified. Its contents are processed with the contents of the list currently been shown by the lists window, and the results go to the currently shown list. If *Copy* is performed, the contents of the process list are simply copied to the currently shown list, overwriting any previous contents. If *Or* is performed, all the addresses in the process list are copied to the shown list, leaving this list with all the addresses that were in either list previously. The two most useful for cross-referencing memory compare lists are *And* and *And not*.

Take a list that contains addresses that changed when the thing you were looking for changed. Choose a list that contained addresses that changed when the thing you looked for changed as the process list, and click on *And*. Do this for all such lists. Then do the same, instead using the *And Not* button, for all the lists containing addresses that changed when what you were looking for *didn't*. The final list should contain just a few addresses. You can now look through them by hand, using the disassembler, to see if they hold reasonable values, and whether they do change accordingly when what you are looking for changes. Once you have found a single address that holds what you are looking for, you can look for code that affects it in the same way as with the normal searching method.

You can save lists out to disc. To do so, choose the list you want, click Menu on the lists window, go over the *Save* item, and save in the normal manner. To load the list back in, drag it to the lists window or the Desktop Hacker icon bar icon. A free list number will be found for it automatically.

Also on the lists menu is the option to *print*. You can print the contents of a list, by going over the *Print* item, choosing the number of copies you want, and clicking on *Print*. You do not need to load the printer drivers, but doing so allows background printing. You can also print a section of the game, as a disassembly. Open the *Print* window from the main menu, choose how many copies to print, give a start and end address to disassemble between, and click on *Print*.

Creating Cheat Modules

To create a cheat module, you must load the *!CheatMod* application. The editor window automatically appears. First, fill in the name of the game, your name, and any other help text that might be necessary to describe how to use the cheat. If this string is particularly large, or there are a lot of cheats in the module, set the *Clear screen* option, so that some of the information doesn't scroll off the window when the cheat module is run.

Next, fill in a check string and address. When any of the cheats in the cheat module are used, this address is checked. If it does not contain the text you type, the cheat is not activated. This protects programs other than the game from being hacked accidentally, and is a good idea to prevent them from crashing, but is not compulsory. A good way to find a check string is to search for the name of the game using Desktop Hacker's text search, then put one of the addresses into CheatMod.

Now add the cheats. Each cheat is activated by holding down *Alt* and pressing a key. Click Menu on the editor window, and choose *Add key*. Put the key that activates the cheat at the top, and give the cheat a name, so that the user can see what it does. Each *Key* may activate many *effects*, each of which alters one or many addresses in a fixed way. To add a new effect, click Menu on the *Key editor* window, and choose *Add effect*. The *Effect editor* window appears.

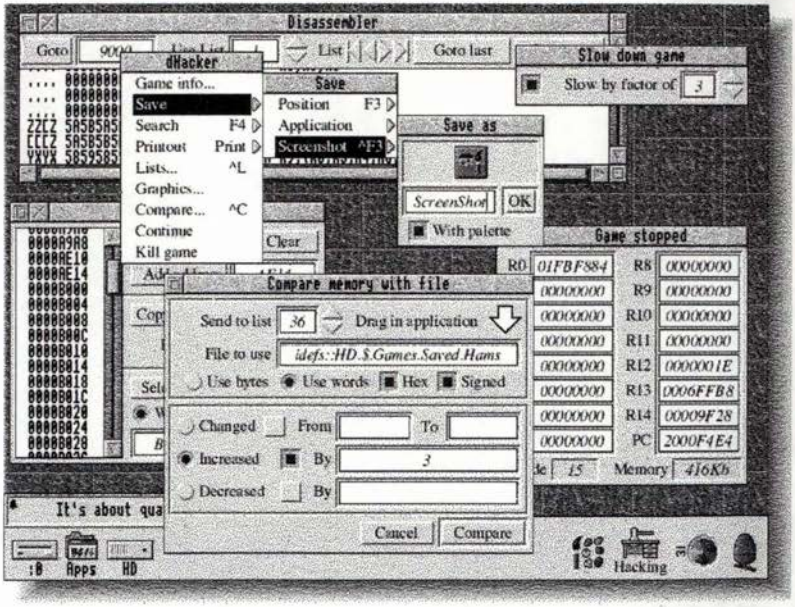
Choose whether to use bytes or words (use whatever you used when searching), type a value (selecting the hex icon to the left of the value if it is in hex), and what action to use: *Load* simply puts the value you specify into the address(es); *Add* and *Subtract* increase and decrease the contents of the address(es) by the amount you specify. *Or*, *And* and *Eor* are bitwise logical operations, and are unlikely to be useful.

Then add the address or addresses, one by one, that are to be affected. Click Menu on the *Effect editor* window, move over the *Add address* item, and type the address.

Many CheatMods may have only one key, with only one effect, affecting only one address. However, the extra flexibility is often needed. To see some examples of programming cheatmods, drag one from the *CheatMods* directory to *!CheatMod*. You can also import CheatMods made by old copies of *The Hacker*.

If you do make any CheatMods of new cheats, please send them to us to include in the collection!

- Stop games at any time and return to the desktop; you can run all your normal applications, in the middle of a game.
- Working on the desktop, Desktop Hacker is by far the most intuitive and easy-to-use games utility. !Help is supported verbosely!
- Find extra or even infinite lives, energy, time, or firepower quickly and easily, with the comprehensive and flexible searching options.
- No experience of ARM code required! However, for the expert, there's a comprehensive disassembler.
- Slow the game down to a fraction of its normal speed!
- Save out screen-shots and rip graphics out of the game!
- Create stand-alone cheats to give to your friends - there are already at least 642 cheats for at least 157 games included free!



DoggySoft

Furzefield House
Furzefield Road
Beaconsfield
Buckinghamshire
HP9 1PQ

Design, coding: Andrew Clover
Phone: (0494) 431916
Fax: (0494) 675878
Data: (0494) 681711
Telex: 83675 Brit G
E-mail: ajc@doggysft.demon.co.uk