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 comand 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

DEEP TRAINER *

There are several trainer type commands introduced with the MKIII Action Replay. These go under the generic title of deep trainer commands. These are totally separate to the above trainer commands and are treated as a different subject. To use the deep trainer you must have at least 1 MB of memory. This technique is of use when the previous commands fail and especially where energy bars are used although it can be *very* time consuming and at first may seem a little bewildering.

TDS *
Deep trainer start

This command starts the deep trainer mode. Note how there are no variables in the line, it is quite simply TDS. You will use this command when you have just started a game, when you have full energy or lives. Each time you freeze the game and the value is the same (full energy 3 lives or whatever) you will use the TDS option.

TDC *
Deep trainer change

This command is similar in a way to the T command. You enter this when the original value entered has changed. So if you started a game with, say, full energy you would start using the TDS command. Then you would lose some energy, re-freeze and enter the TDC command. Notice how no numbers are entered. From here on you keep re-freezing the game at different points. If you have full energy you will use the TDS command, if you have anything but full energy you will use the TDC command.

TDD (start) (end) *
Deep Trainer Delete Addresses

This command is for the more experienced hacker. It will remove all search addresses within the range (start)-(end).

TD *
Display deep trainer addresses

This command will display a list of all the addresses under investigation by the deep trainer. If you try it after just starting with the TDS command the list will be very long!

TDI *
Display Probable Deep Trainer Addresses

This is the most important of the deep trainer commands. It will list all possible locations that your energy/lives etc. can be at. You should repeat the TDS and TDC commands over and over until typing the command TDI comes up with the same values over and over again. One of these values is the address you are looking for. If you find that there are no possible addresses it is more than likely that you have made a mistake and typed TDC instead of TDS or vice versa. You should TDX and then start again.

TDX *
Exit Deep Trainer

This command is used to abort the previous deep trainer commands. All addresses will be reset. This is useful when you have made a mistake and need to restart.

USING THE DEEP TRAINER

There follows one or two examples on how to use the deep trainer. Even if you do not have the games mentioned you should try to follow what is happening as it will help you to understand the principle of the deep trainer.

ROBOCOP2

The following procedure will find the life counter on Robocop2. First switch the machine on and go to the F3 options screen. Switch off all extra memory, i.e. select 512K chip mem only and no extra fastmem. Load the game from disk and when loading is complete start the game. When the game is started press the freeze button. Now type the following lines

```
TDX
TDS
```

and return to the game. Lose one life and refreeze the game. Now type the line

```
TDC
```

Lose a further life and refreeze the game and type the line

```
TDC
```

again. If you lose another life the game will end. You should restart it and freeze again. You now have 3 lives which is the same as when you first used the TDS command so you should again type the line

```
TDS
```

and then repeat the procedure above i.e. lose life, TDC lose life and TDC again. After this procedure type the line

```
TDI
```

on my machine the list of possible locations was as follows

```
000B02  000C34  008035  00C689
```

you may find on your machine one or two extra, this depends at which points you had frozen the game. Only one of these values is the correct one so a technique of trial and error is called for.
If you try a TFD command (used in detail in the previous trainer section) on each of these values nothing will be found, however if you use the technique of using one less with the odd numbers on 8035 i.e. 8034 as follows

```
TFD 8034
```

Action Replay will find and eliminate anything that decreases your number of lives.

9

TRAINER

Those of you that have read the previous section will note that this address is the same as that found in the previous section only it took longer. The beauty of the deep trainer method is that no values are needed and therefore you can use it with energy as well as lives.

10

MISC COMMANDS

RAMTEST (start) (end)

Check Memory for Faults.

This instruction will destroy all data between (start) and (end). It writes 0 to all locations in the region (start)-(end) and reads them back, then does the same for FF (hex).

PACK (start) (end) (dest) (crrate)

Pack Memory

This command will pack the block of memory between start and end and place it at (dest). (dest) may be at (start) if required, and uses the compression rate (crrate) as explained in the SA command. When completed the length of the compressed data will be displayed, e.g.

```
PACK 40000 4FFFF 50000 !200
```

will pack all data between 40000 and 4FFFF and place the compressed data at 50000 using a compression rate of 200 decimal. Make sure you keep a note of the length!

UNPACK (dest) (end of packed)

Unpack Packed Data

This is the reverse of PACK. You must specify the destination of the unpacked area (NOT OVER ITSELF!) and the last byte of the compressed data (end of packed) which can be calculated by the start of the compressed data + length displayed after completion of the PACK command. For example assuming the length of the packed data in the previous command was 1234 then the reverse of the above example would be.

```
UNPACK 40000 51234
```

COLOR (back) (pen)

Set Screen Colours

Will set the screen colours using the Amiga palette (0-!4095) (back) being of course the background colour and (pen) the foreground colour. This is an alternative to the preference screen (F3). When used on its own (COLOR) it will display the background and foreground settings.

RCOLOR
Reset Screen Colours

This command is for when you have made a mess using the COLOR command.

TMS (address)
Mark Address

This command is used as a notepad for reminding you of an address you are currently investigating. They are also saved using SA and reloaded using LA/LR so you can remember for example where an infinite lives location is. There are 10 possible memory markers.

TMD (address)
Marker Delete

This will delete one of the current memory markers

TM
Show marker

Will display the current memory markers.

SPR (nr|addr)
Edit Sprite

Will edit the sprite number (nr) or the sprite at address (addr). If, for example, we have frozen workbench we type the line

```
SPR 0
```

and press (return) about twenty times, the outline of the pointer would appear in various numbers, each number corresponds to a colour. We can if we like move the cursor keys

and edit the sprite making sure we press return after every line we have changed. Note you can only use the numbers 0-3 as you can only have 4 colours per sprite.

VERSION
Show ROM Version

Will display the date and version number of the current ROM's

MEGASTICK (1) *
Joystick Handler

This command invokes a new screen - the joystick handler screen. It is used so you can set the joystick up to act as certain keys, e.g. type megastick, press the fire button on player 1 joystick, pull to the left and press space. From now on when you restart, moving the player 1 joystick to the left and pressing fire will produce a space. All directions can be set up and combinations of fire-pressed etc. Press escape to accept a setup. Use the option 1 if you only wish to select player 1.

NOSTICK *
Remove joystick handler

Disable the mega stick feature.

CLRSTICK *
Clear megastick values

This command clears any codes set by the mega stick command back to zero.

SSTICK *
Save megastick values

Saves the values that have been set up using the megastick command to disk for re-use at a later date.

LSTICK *
Load megastick values

Loads the values for the megastick command that have been saved using the Sstick command.

RESET *
Reset Amiga

Resets the Amiga as though ctrl-A-A has been pressed.

PAL *
Switch to PAL mode

Switches the Amiga into the Eurpoean PAL standard.

NTSC *

Switch to NTSC mode

Switches the Amiga into the American/Japanese NTSC mode.

SETMAP *
Set keymap

Will display a screen which will (if possible) allow the editing of keys on the Amiga keyboard. For example the function key 1 can be changed to send the words "Function key pressed". To use this command simply type setmap. Use the mouse to select which key you wish to change and click on it. A new window is displayed with the current string and allows you to change it. When you have entered a new value press (enter) and click on (OK) to accept. When you are happy with the new keymap you can save it by clicking on the appropriate option for reloading at a later date. When you wish to install the new keymap click on (install) and the new map will be written to memory so when you exit Action Replay the new map will be used.

ASCII *
Display ASCII map

This deceptively useful command will show the ASCII (American Standard Code for Information Interchange) character map and the corresponding codes. It is ideal for programmers who do not have a reference manual to hand.

ALERT (guru-number) *
Display Guru List

This command is useful for programmers - when you get the guru numbers it is often difficult to remember what the function is. For example the guru exception 8400000C does not mean much to me but if you type the line

```
ALERT 8400000C
```

will reveal all (well maybe if you are a programmer). Typing Alert on its own will list all the gurus and descriptions of such.

SMALLCHAR *
Small Printer characters

If you have an Epson printer attached this command will reduce the size of the text sent to your printer.

NORMALCHAR *
Normal Printer characters

Reverses the action of smallchar.

PRT *
Print String

This instruction sends a sequence of characters to the printer. Either text in quotation marks may be sent or a number may be sent in which case the ASCII value is sent to the printer. The following example will print "hello there" in expanded print to an Epson printer.

10 MISC COMMANDS

PRT 1B 57 31 "hello there"

VIRUS
Search memory for virus

Will search through memory for any known virus. This is not usually necessary if you have left the virus test ON in the (F3) preference screen as you will be warned of any virus. Please note if you have any virus killers in memory this may confuse Action Replay as the program will have bits of the viruses' DNA in memory.

KILLVIRUS
Search and Remove Virus

Search for and remove any viruses resident in memory.

11 MONITOR

The following set of commands are of most use to machine code programmers. Again I must stress that to beginners they may seem completely bewildering. However, if you are interested in programming machine code there are numerous Amiga books that could start you off and as you read the Action Replay monitor commands, they will begin to make sense. For those of you that are already conversant in machine code you will find that these commands will become essential and you will wonder how you ever did without them. One or two significant additions to the MK II have been made to make life easier, such as the lm command and the trace commands.

SETEXCEPT
Sets The Exception handler

Gets rid of those annoying guru messages.

COMP (start) (end) (dest)
Compare Memory Blocks

Will compare the memory between (start) and (end) with that located at (dest). All diferences in the destination block will be displayed.

LM (path)(name),(dest) *
Load file to memory

This allows machine code or data to be loaded straight into memory at any point and be investigated at leisure. The memory remains when the program is restarted so is ideal for changing blocks of data/graphics etc. e.g.

LM DATA,70000

will load the contents of file data into memory starting at location 70000 in hexadecimal.

SM (path) (name),(start) (end)
Save Memory To Disk

This command is used to save a block of memory to disk, in the standard (path)(name) format,which is located between (start) and (end). Ideal for saving blocks of machine code or other forms of data, e.g.

```
SM scroll,3021F 312EA
```

SMDC (path)(name),(start) (end)
Save Memory To Disk In DC.B format

This feature is ideal for the assembly language programer where a block of memory can be saved out as ASCII in the style DC.B followed by the byte values separated by commas which can be loaded into most assembly packages such as DEVPAC, ARGASEM etc.

SMDATA (path)(name),(start) (end)
Save Memory To Disk In DATA format

This feature is similar to SMDC only the format is DATA instead of DC.B

A (address)
Start M68000 Assembler

Begin entering assembly language instructions from the specified address, all standard 68000 instructions are supported and as long as you enter valid instructions you can keep on entering. To finish simply press (ESC). I am *NOT* going to explain the 68000 instruction set, there are plenty of good books on that.

BS (address)
Set Breakpoint

This command is ideal for the hacker or programmer who requires information at a certain point in a program. You can set a point in memory specified by (address) to break out of the main program and enter Action Replay. You may use this instruction after using the TF command and finding a life decrement instruction, say at location 41259. So you could investigate in more detail the screen and registers, use the commands

```
BS 49152
X
```

and as soon as the location 49152 is reached you will get the Action Replay screen back.

B
Show Breakpoints

Will show all breakpoints set using the BS command

BD (address)
Breakpoint Delete

Delete the breakpoint previously set at memory location (address).

BDA (address)
Delete all breakpoints

Remove all breakpoints that have been set using the BS command.

X
Restart current program

This will quite simply start the program from where you pressed the freeze button. Any changes you made to the registers or memory will be kept and so if you have made major changes the computer may produce unexpected results or crash.

C (1|2|address)
Copper Assembler/Disassembler

The line that should be entered is C then 1 or 2 or an address. The Amiga is equiped with a copper or co-processor which can be programmed in a similar, but more limited fashion to the 68000. After you enter the C and one of the possible options the copper instructions of the locations will be displayed. Pressing (return) will enter the instruction on the line and move to the next instruction in memory. For information on the copper chips please consult an Amiga Technical reference manual.

D (address)
Start Disassembly

D followed by the (return) key will begin Disassembly in 68000 m/c from the point at which the program was frozen. If an (address) is entered then dissassembly is started from that point. Please note you can specify numbers in different ways e.g. T0 for track 0 if the diskmonitor is being used or \a0 etc. for the machine code registers, for example

```
R
```

the computer may respond with

```
D0=FFFF1234 00000000 00000028 000059d2 etc.
A0=00000676 000FFE1E 000FFF75 00004984 etc.
```

to view the machine code starting at the address in A1

```
d \A1
```

will start dissassembly from address FFE1E, pressing (return) will dissassemble the next line and so on until (ESC) is pressed. The cursor keys can be used to control the direction of assembly.

E (offset)
Show/Edit Chip Registers

Will show the contents of the chip registers in binary. These values can be changed using the cursor keys and pressing (return) when a line has been modified.

F (string),(start)(end)
Find String In Memory

Will search through memory between (start) and (end) for all occurrences of (string), which should be surrounded by quotes. The search is case sensitive. This command

is also useful for finding occurrences of bytes in memory or even instructions. FS will search for a string but is not case sensitive e.g.

```
F "COMMODORE",0 7FFFF
```

will search for COMMODORE (note in upper case) between the limits 0 and 7FFFF.

```
F 4E 71,70000 80000
```

will search through memory from hex 70000 to hex 80000 for occurrences of bytes 4E followed by 71 which also corresponds to the assembler instruction NOP.

FA (address) (start) (end)
Search for Adr Addressing opcode

Will search through memory for Adr Addressing opcodes. This is a very useful debugging/hacking tool for the more advanced user. Its action is to search through all locations between (start) and (end) for machine code instructions that access location (address). FAQ will fastsearch through memory. The use of these instructions is to find out where a subroutine is called from, e.g.

```
FA 71000 60000 80000
```

on my machine responded with the following lines

```
070002 BSR   00071000
07000C BRA   00071000
Searched up to adr: 080000
```

FR (string),(start) (end)
Search relative for String

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

G (address)

Goto Address

G on its own will act the same as X. With an address specified G will set all the registers to what they were when the program was frozen (unless they have since been modified) and jump to (address). Ideal if you want to redirect the program or test out an Assembly program you have just entered.

TRANS (start) (end) (dest)

Transfer Memory Block

Will transfer a block of memory between (start) and (end) to Destination (dest). The transfer is intelligent so you may specify a (dest) which lies between start and end.

WS (string).(address)

Write String To Memory.

Will write a STRING (bounded by quotes) to the memory starting at (address) you can also use numerical values (not in quotes), e.g.

```
WS “HI There”,41259
```

M (address)

SHOW/EDIT Memory As Bytes

This will display a line of data in byte format in Hexadecimal starting from location (address). The bytes can be changed to any valid Hexadecimal number using the cursor keys and approriate numbers. When a line has been altered pressing (return) will enter those alterations to memory and display the next set of bytes from memory. Press escape to exit this feature.

N (address)

Show/Edit Memory As Bytes

Similar to the M Command but memory is shown as ASCII.

NO (offset)

Show/Set ASCII 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, e.g. if we did an ASCII dump using N and got the following in a block of text

```
IFMMP
```

and we then entered the line

```
NO 1
```

The same piece of text viewed using another N command would be

```
HELLO
```

So the command can be used for showing hidden text. NO on its own shows the current (offset).

NQ (address)

Display Memory Quick

Will display memory in ASCII in quick format

MEMCODE (start) (end) (code)

Encode Memory

Will go through a block of memory encrypting each byte using the 68000 EOR instruction with (code). This has the effect of rendering the block unreadable. To decypher this code you may either use the 68000 instruction EOR.B or execute the instruction again in an identical manner e.g.

```
MEMCODE 312FE 345A1 EA
```

will encode a block of memory using the key EA (in hex). To get it back to normal simply type

MEMCODE 312FE 345A1 EA

O (string),(start) (end)
Fill Memory With String

This command will fill a block of memory with repeated sequences of (string), which should be bounded by quotes.

R (register) (value)
Display/Modify Registers

R (return) will display the contents of all the 68000 registers at the time of freezing. If you wish to modify these registers you should specify both a (register) number and the (value) to be inserted in it e.g.

R D0 !1000

will insert 1000 decimal into the 32 bit data register D0. Any modifications to these registers will become valid when you either exit using the X or G commands.

W (register)
Display CIA Contents

Displays the contents of both the CIA (8520) chips. The value in (register) used to define the offset from the start of the CIA's. Both chips are displayed on the same line, the first and second numbers being the first and second CIA's correspondingly, for example

W 4

will display the contents of the fourth register on both the first and second CIA's.

Y (address)
Display memory in binary form

This instruction is very similar to the M command although memory is displayed and edited in binary form. The amount of bits displayed on one line is specified by YS (value) where value is 1-8 giving the amount of bytes displayed per line.e.g.

YS 8

will tell the Y command to display 64 bits per line. The address will be updated accordingly.

MS (address) *
Set memwatch point

This is a wonderful instruction for all you programmers out there - how many times have you found memory locations becoming corrupted and wondering what on earth was responsible for doing this? All you need to do is use the MS command with the byte address you wish to keep a check on and start your program (either change register PC or the G or X commands) and as soon as any command changes your location, Action Replay is activated and the program counter points to the next instruction after the one which has done the damage, e.g.

MS 70000

would, as soon as any new value is entered into location 70000, freeze the program and display a message. This I find is of more use than the breakpoint instructions.

MW *
Show watchpoints

This will display a list of all the memory locations set using the MS command.

MD (address) *
Delete Mem watchpoint

Will delete an address from the memwatch list.

11 MONITOR

MDA *
Delete All Watchpoints

Will remove all addresses set using the MS command.

TR (steps) *
Trace without subroutines

This is another useful command, it will trace a machine code program (steps) number of steps from the current frozen position (defined in the registers). All memory and system data is updated including screens etc. This makes this instruction far more powerful than equivalent disk based monitor trace commands as all system information can be viewed, e.g. the screen using PC. Note that all subroutines are executed as a single instruction.

ST (steps) *
Trace with subroutines

This is similar to the TR command apart from the one difference in that subroutines are stepped down as continuous order and are not treated as a single instruction as is TR.

? EQUATION *
Calculate Value

This instruction is ideal for altering between number bases. The result of the (equation) will be displayed in binary Hex and Decimal e.g.

```
?8+7+!15*2
```

will give the result !60. Notice this is not the same as previous versions of Action Replay, the standard priority orders of multiplication and division first and addition and subtraction second are now used. Valid symbols are (+,-,/,*)

12 COMMANDS FOR SYSTEM INFORMATION

Most of the commands in this section are devoted to complex System information. Some, however, such as Avail are useful to everyone.

AVAIL
Available Memory

Quite simply displays the memory available on the machine both chip ram and Fast ram. Ideal if you have fitted one of DATEL's Rammaster II's for example to check that it is functioning properly.

INFO
Display System Info

Will display a list of important system information for example the colour palette values.

LIBRARIES
Show Execbase Librarylist

This command will display a list of the Execbase Library list and their corresponding addresses.

INTERRUPTS
Show Execbase Interrupt vectors

Lists all currently active Execbase interrupts.

EXCEPTIONS
Show Exception and Interupt vectors

Shows the list of 68000 exceptions and also shows where in memory they are currently pointing.

EXECBASE

Show Exception and Interupt vectors.

12

COMMANDS FOR SYSTEM INFORMATION

RESOURCES

Show execbase Resource List

CHIPREGS

Show Name And Offset Of Chipregs

DEVICES

Show Execbase Devicelist

TASKS

Show Execbase Tasklists

PORTS

Show Execbase Portlist

INDEX

INDEX

NOTES