QDBug

The definitive debugger

```
.fun
ADR R1,clain:MOV R0,#&1D:MOV R2,#2:SWI "XOS_Claim"
ADR R0,ssp:MOV R1,R0:SWI "XOS_ReadVduVariables"
MOV R0,#0:MVN R1,#1<<10:SWI "XOS_ChangeDynamicArea";;BUG release
MOV R0,#3:MVN R1,#1<<10:SWI "XOS_ChangeDynamicArea";;BUG release
MOV R0,#4:MVN R1,#1<<10:SWI "XOS_ChangeDynamicArea";;BUG release
MOV R0,#5:MVN R1,#1<<10:SWI "XOS_ChangeDynamicArea";;BUG release
MOV R0,#200:MOV R1,#3:SWI "XOS_Byte"
LDR R1,ssp:RSBS R1,R1,#&14000*6
MOV R0,#2
SWIGE "XOS_ChangeDynamicArea";BUG release
MOV R4,R1
ADR R1,clain:MOV R0,#&1D:MOV R2,#2:SWI "XOS_Release"
\MVN R14,#8:\LDR R14,[R14]
SWI &116:SWI &100
CMP R0,#&101:MOVCS R0,#100
SWI "XOS_GetEnv":MOV R13,R1
ADR R0,p_in:LDR R1,p_scpos:SWI "XOS_ReadVduVariables"
LDR R0,[R1]:LDR R1,e1:MOV R2,R1:MOV R3,R1:MOV R4,R1
MOV R5,R1:MOV R6,R1:MOV R7,R1:MOV R0,R1
.e1s STMIA R0!,{R1-R8}:CMP R0,#&2000000:BLT e1s
\LDR R0,[R0,-R14]
LDR R1,p_scpos:LDR R2,[R1]:ADD R10,R2,#&8000:ADD R8,R10,#&14000
.load_spell
```

Produced by Vertical Twist
Program written by Christophe Thivend, ArcAngels Software
Marketed by Leading Edge

Contents

QDbug

Introduction.

QDBug has been written to help programmers find the bugs that almost invariably creep into code. The 'professional' and ARM code learner can benefit from using this program as it allows you to single step though code looking for where things go wrong.
It has been written to remain almost invisible until needed, when with the press of the 'hot keys' QDBug will come to life.

There are many features including : simple window based editing system, BASIC source code can be viewed during execution, single step and skip instructions, REPEAT-UNTIL,WHILE and DO loop execution, alter memory between processor instructions, multiple break points, search fill and copy memory blocks with the added facility of remote terminal control. All this and much more.

Getting started.

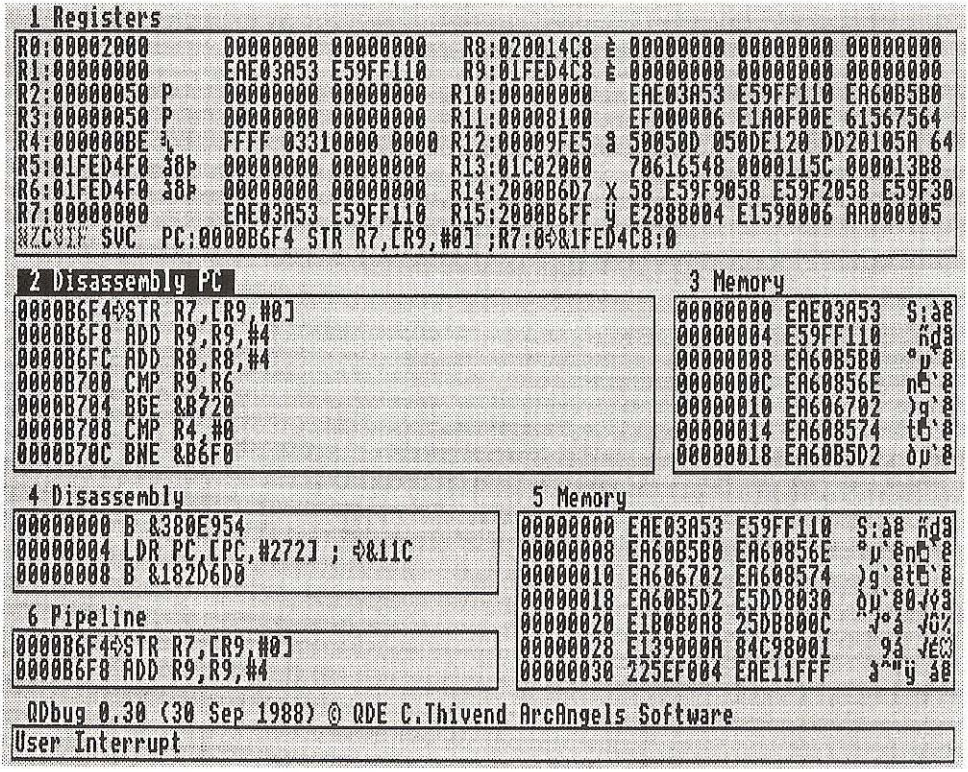
Loading QDBug
First load QDBug by double clicking on its icon from the disks filer window. A small box will appear briefly to confirm that it is installed.

Activating QDBug
You can now enter QDBug at any time simply by pressing [SHIFT]+[Alt]+[Alt]+[SHIFT] keys. The main control and editing window should now appear.
There are also 2 other ways to enter QDBug, these methods are detailed later in the manual under '*Break' and 'SWI calls'

Exiting QDBug
To leave the QDBug editing window, press the [Ctrl]+[R] keys. This is the 'Run' command. There is also another way to exit QDBug this is detailed later under 'OS_Exit'.

QDBug info
Press [Ctrl]+[F6] to see extra information on your copy of QDBug.

Using the QDBug windows



Overview

Above is a screen shot of the main QDBug control screen. As you can see it is made up of a number of separate titled boxes. Each of these 'boxes' or windows can be either : moved, have their size changed, be deleted, or their function changed. This allows you to customise the screen to your own requirements. These changes can also be saved to a file for repeated use.

Up to 8 windows can be displayed at once but overlapping ones are not supported. They are also updated in real-time but this stops momentarily when a key is pressed.

Each of the windows are numbered and titled for easy recognition.
Window 1 always displays the register contents and cannot be deleted or altered.
Window 8 displays information about the last thing that occurred. This leaves windows 2 to 7 which are free for the user to set up as required.

These windows can be of 3 types : Disassembly, Memory or Pipeline. These can also have a "window start address", which is the pointer to memory to disassemble or to dump from.

There is also be a "start address locker", which is an expression that is calculated each single-step and the result can become the "window start address". Once your perfect window setup has been achieved, you can save the configuration to disk to be used again.

Selecting a window
The [Tab] key changes the currently selected window. A black box will appear around its title to confirm that it has been selected.
[Shift]+[Tab] changes the selected window in the opposite direction to [Tab].

Altering windows
To change the shape or position of any of the windows 2 to 7 press [Ctrl]+[F9]. The keys [M] and [S] change the effect that the cursor keys have on the window selected.

QUICK KEY FIND : [Ctrl]+[F9]
KEYS : [S] - Stretch
[M] - Move

Press 'Escape' to exit this window altering mode.

Clear screen.
When windows are moved around, sometimes 'debris' is left on the screen. This can be cleared by pressing [Shift]+[F11]. This is also useful if you are using a terminal as the terminal screen will also be cleared.

QUICK KEY FIND : [Shift]+[F11]

Delete Window
If you do not need a window then it can be deleted. Use [Shift]+[F9] on the selected window. This can only be done on windows 2 to 7.

Adding a window
When an extra window is needed, it can be created by pressing [Alt] with a number between 2 and 7. This number represents the window number to be created.

QUICK KEY FIND : [Alt] + [2] to [7]

Window Start Address
Once selected (with [Tab]) a window can be given a pointer in memory from where it should start its memory dump or disassembly.

To change the Window Start Address press [Ctrl]+[F12]. You will then be prompted for an address, enter your choice then press [Return].

QUICK KEY FIND : [Ctrl]+[F12]

Window Start Address locker
Each window can have a start address that is calculated from an expression. An example would be 'PC' or program counter. Press [Shift]+[F12] to enter this option. You will then be prompted for an expression. Please refer to Appendix A for the list of available expressions.

QUICK KEY FIND : [Shift]+[F12]

Execute Window Start Address Locker
Once a 'Start Address Locker' has been defined it can be 'Executed' so that the result can be stored in the 'Window Start Address'. To do this press [F12]. This is useful if you have used a window to look else where in memory and now need to return. Select the window you are interested in first.

QUICK KEY FIND : [F12]

Printing windows
If you wish to print the contents of a selected window then press [Ctrl]+[Print]. As QDBug uses its own internal printer driver, a RISC OS one does not have to be installed. You must first set up the printer using the printer options.

QUICK KEY FIND : [Ctrl]+[Print]

Printer Options
Various printer options can be selected so that compatibility can be achieved with a number of different printer types. To select 'Printer Options', simply press [Shift]+[F3] then [P]. The 'New Line' option tells the computer which sequence of bytes to transmit for a carriage return.

The Light On and Light Off options refer to the sequence of bytes that control superscript or condensed text on your printer.
For each of these options you can enter up to four bytes.
Use the cursor keys to select your option and 'Escape' to exit.
If you have a RISC OS printer driver installed then select the [R] option.

QUICK KEY FIND : [Shift]+[F3] then [P]
KEYS : [R] - Risc OS printer driver, on/off

Changing character height
Sometimes it may be necessary to change the height of the characters in a window. First select a window then press [F10]. The height will toggle between normal and double height.

QUICK KEY FIND : [F10]

Changing Window Type
When you are setting up your own QDBug screen, you may wish to change the type of window that you are using. There are three choices : Disassembly, Memory dump and Source. As you press [Shift]+[F10] the window type will cycle between the options.

QUICK KEY FIND : [Shift]+[F10]

Returning to the main control window
Press [Escape] at any point to return to the main control window of QDBug.

QUICK KEY FIND : [Escape]

Full screen
Press [Home] to make the currently selected window go to full screen size.

QUICK KEY FIND : [Home]

Cancel full screen
Press [Copy] to cancel the action of [Home].

QUICK KEY FIND : [Copy]

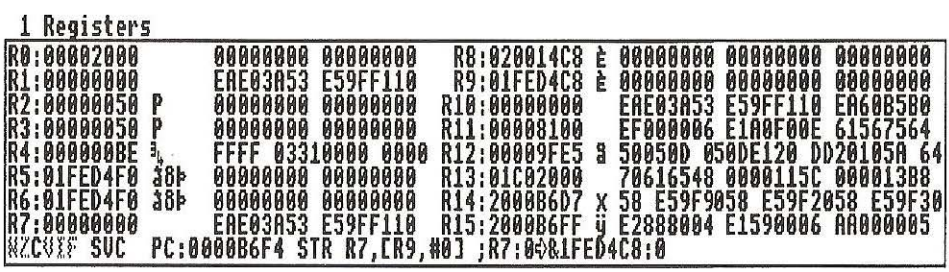
Instant window selection
To select a window directly press [Alt]+[1] to [Alt]+[7]. If a window is closed then it will be re-opened.

QUICK KEY FIND : [Alt]+[1] to [7]

Different window types

The register window

The register window displays the contents of all the 16 ARM registers as well as the condition of the status registers. It also shows the processor mode and next instruction that will be executed.



This is always referred to as window number 1 and cannot be moved, stretched or deleted. You can however change the height to double or normal height with [F10] once it has been selected with [Tab].

Status Flags : These are displayed on the bottom left as a series of letters, N,Z,C,V,I,F. When they appear black that means that particular status bit is set.

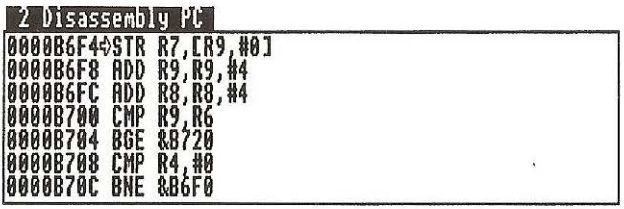
Processor mode : This is also on the bottom left of the window. The possible modes are : User mode, Supervisor mode, IRQ mode and fast interrupt mode (FIQ). These are represented with : USER, IRQ, FIQ and SVC

The value of the PC (or program counter) is displayed at the bottom of the window. The PC holds the address of the next instruction to be executed. It is derived from R15, where PC=(R15-8) BIC &FC000003. After the PC display a disassembly of the first instruction in the pipeline is shown.

Extra information is provided after the disassembly, depending on the type of instruction being executed.

For example, "R6<&1C" means that R6 will take the value &1C
"R14:&285<&2568:&8010" means that R14, which has at present, the value &285, will take the value &8010 from the address &2568

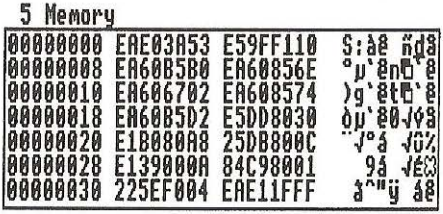
Disassembly window
This window displays a disassembly of memory from its 'Window Start Address' as described earlier in this section. See also 'Window Start Address Locker'.



The small arrow shows the position of the instruction being executed, it sometimes changes direction. When the processor is forced to jump to a new location the arrow will show the direction of the jump.

Associated functions : 'Alter window', 'Start address', 'Set Address Locker', 'Execute'

Memory dump window
This type of window displays a memory dump from its 'Window Start Address'. For more information, see the section earlier on window start addresses.



Editing memory using a memory dump window
If you press *[F11]*, a cursor will appear in the currently selected memory dump window. You can now edit memory locations using the cursor keys to scan though memory. *[Tab]* will change between hex and ascii, and *[Escape]* will exit.

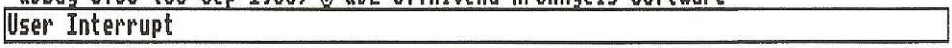
QUICK KEY FIND : [F11]

Different memory dump types (Intel and Motorola)
Different types of memory dump can be selected, these are 'Motorola' and 'Intel' type dumps. The ARM uses the Intel method of storing code in memory.

This function changes the memory dump in ALL windows, including the Register window. Intel type memory dump is selected by [Ctrl]+[F11], and Motorola mode is selected by [Ctrl]+[F10].

QUICK KEY FIND : Intel - [Ctrl]+[F11]
Motorola - [Ctrl]+[F10]

Current Status Window
This is the window at the bottom of the QDBug main screen however it cannot be altered in any way. It shows you the current QDBug status. For example when you first enter QDBug with the 'hot keys', the status is "User Interrupt".
Address exceptions and Data aborts are also reported here.



Pipeline window
This window type shows the instructions that are in the processor pipeline.

The small arrow points to the actual instruction being executed. When it changes direction it means that it is pointing in the direction that the processor is about to jump.

QDBug Workspace

Workspace

QDbug needs workspace in the RMA for some of its functions. These include macros, labels, blocks, breakpoint expressions, source code viewing and to speed up remote terminal communications.

You can allocate an amount of memory when first loading QDbug.
Use the command *QDBug -w xx to do this. Where xx is the size in Kb of required workspace.

If QDbug is already loaded then press [F1]. This accesses the CLI, now use the command QDbug -w xx to change the workspace size. An example size would be 32K.

[F] - displays the amount of workspace reserved and the amount free.

QDBug *commands

***Break**
Break enters QDbug.
To leave, use Ctrl-R.

Trapping named files when they are executed

***Break Filename**
*Break is used to enter QDbug if an absolute file, module or utility of a particular name is being executed. After loading, QDBug is entered, and the PC is set to the first instruction of the program.

This is very useful for debugging RiscOS application. This is because you can set up breakpoints on Wimp_Poll or anything else inside an application.
The syntax is "*Break FileName" where filename does not contain any path name and is not inside quotes.

For example:
**Break !RunImage*
Will break the execution of any program, modules or utility called !RunImage.
You can have only one name defined at any one time.

***NoBreak**
NoBreak is used to cancel the action of *Break.

***QDbug**
Used with the "-w" options, this allows you to set up the size of the workspace.
Warning, you can loose data, if you reduce the workspace by too much.
This can also be used from the 'CLI' inside QDBug.

For example:
**QDbug -w 32*
Will set the size of the workspace to 32Kbytes.

Used with the "-l", you can load a macro file.
For example :
**QDBug -l FileName*

***QDsave**
This saves the labels and blocks to a file.
For example :
**QDsave FileName*

***QDload**
This loads the blocks and labels from a file.
For example :
**QDload FileName*

File handling

Overview

Files can be loaded from disk into either normal memory or QDBug workspace memory. Generally speaking BASIC programs should be loaded into workspace and ARM code, into normal memory. The following functions are described below : Catalogue current directory, Set current directory, Load file, Load source, Save file and Stop Drive.

Current Directory

Pressing [Shift]+[F1] from QDBug displays the contents of the currently selected directory.

QUICK KEY FIND : [Shift]+[F1]

Set Current Directory

If you wish to change the currently selected directory then press [Ctrl]+[F1]. You will then be asked for your directory name.

QUICK KEY FIND : [Ctrl]+[F1]

Load File

This allows you to load a file into main memory, you must enter the file name and load address. If you just enter the file name then QDBug will look at the file type or load and execute address if present.

For example, this means that when loading a backup of memory that has been saved with QDBug, you type in the file name then press [Return] twice.

QUICK KEY FIND : [F2]

Load Source

This allows you to load source programs into workspace memory. There are 32 slots for non-BASIC files and 32 slots for BASIC programs. You will first be asked for a slot number, then a file name. If there is a file already in the slot then it will be cleared and replaced with the one just loaded.

These 32 slots are overlapping. This means that if the program is BASIC then it is automatically loaded into the BASIC slots, and if text based then it is loaded into the text slots.

See the chapter on 'Viewing BASIC source and Text files in QDBug" for more information on this topic.

QUICK KEY FIND : [Shift]+[F2]

Save File

This allows you to save a section (or block) of main memory to a file. You will be asked for Start address, end address or length, then the file name.

QUICK KEY FIND : [F3]

Stop Drive
Sometimes when you jump into QDBug the floppy drive light will remain on. This option allows you to switch the drive off. Just press [Ctrl]+[F2].

QUICK KEY FIND : [Ctrl]+[F2]

Viewing BASIC source and Text files in QDBug

Overview

When debugging, it is sometimes useful to be able to see the original source code. This can be in the form of BASIC programs or Text files. Once loaded, one of the QDBug windows can be configured to display your source.

Loading Source code

To load source code just press [Shift]+[F2]. You will then be prompted for a slot number between 1 and 32, then the file name. Remember to select the correct directory that your file is stored inside using the 'Set Directory' option ([Ctrl]+[F1]).

As these files are stored inside the QDBug's RMA workspace area you may need to refer to the section in the manual called 'QDBug workspace' if memory shortages occur.

QUICK KEY FIND : [Shift]+[F2]

Slots

There are 32 slots for Text files and 32 slots for BASIC programs which are overlapping. This means that if the program is BASIC then it is automatically loaded into the BASIC slots, and if text based then it is loaded into the text slots.

BASIC viewing windows

QDBug windows can be set to display any of the loaded BASIC programs but BASIC V1.04 or 1.05 must be present to use this function. To display the program in a window, select a window then press [Shift]+[F10] until BASIC appears as the window title.

If one of the 32 BASIC slots are free and there is a BASIC program in main application memory (&8000 onwards) then it can be viewed as a BASIC program in a QDBug window. The up / down cursor keys can now be used to view the file

NOTE : The SWI command 'QDbug_SetBlock' for extra relevant information.

TEXT viewing windows

After loading the text select a window then press [Shift]+[F10] until 'Basic' appears as the window title. Next press [Ctrl]+[Shift]+[F10], 'Basic' will now change to 'Source'.

The up / down cursor keys can now be used to view the Text file.

Setup Options

Overview

QDBug can be made to behave differently from its default setup, things like screen mode, execution options, dump control, IRQ handling, Terminal setup and options saving / loading can be accessed from this menu.

Press [Shift]+[F3] to show the 'Options' menu. As with other QDBug menus, you can now move the small arrow to your selection where you can then press a letter that will correspond to a particular option, ie [D] for [D]ump.

[Escape] will return you to the main control screen.

QUICK KEY FIND : [Shift]+[F3]

Dump Options

Use the up / down cursor keys to chose the option you wish to change.

QUICK KEY FIND : [Shift]+[F3] then [D]

SWI Display type - This option controls the way QDBug displays SWI calls on the disassembly window. The options are : Hexadecimal, Internal table and to call the RISC OS OS_SWINumberToString SWI. The last option is generally the better one.

KEYS : [H] - Hex
[I] - Internal table
[C] - Call RISC OS_SWINumberToString

Stacks - If a register is set, then the disassembly of LDM will display'EA', 'ED', 'FA', 'FD'. Otherwise, 'IA', 'IB', 'DA', 'DB' will be used.

KEYS : [0] to [F] toggle on / off

Number format - This controls the display of numbers. You can choose decimal, hex or both formats.

KEYS : [H] - Hex
[D] - Decimal

Disassembly information - This controls the additional information that can be displayed in a disassembly window. You can switch it on or off with the [D] key.

KEYS : [D] - Extra information toggle on /off

Memory Display Type - This controls the type of memory dump in all the windows. It is also equivalent to pressing [Ctrl]+[F10] or [Ctrl]+[F11]. You can choose Intel or Motorola type dumps.

KEYS : [I] - Intel
[M] - Motorola

Branch Info Window - With this option, you can select a window to see the instructions that could be executed by a B or BL instruction. Select a window number for this function, and when a branch instruction occurs that window will change from its original function and display the possible branch code as a disassembly window.
The 'Standing' option means that the window will remain a disassembly window after a branch instruction and its start address is set to the destination of the PC.
The 'Punctual' option means that the window will recover its old status, type and address after the next key press.
KEYS : [O] - Branch Info Window, toggle on / off
[2] to [7] - window select
[S] - Standing mode
[P] - Punctual mode

Tab Length - You can choose values between 1 to 9.
KEYS : [1] to [9]

Execution Options
Use the up / down cursor keys to chose the function you wish to change.
QUICK KEY FIND : [Shift]+[F3] then [E]

SWI Emulation - During single stepping, you can select whether QDBug single steps though SWI calls or it sets the PC to &8 (in SVC) and calls the SWI. These modes are known as 'Emulate' or 'Follow'.
KEYS : [E] - Emulate
[F] - Follow

Running Control - Running control enables is the following functions : Ctrl-W, Ctrl-Q, Ctrl-R, Ctrl-T, Ctrl-S, Ctrl-F, Ctrl-L, Ctrl-X, F5 and Shift-F5.
KEYS : [E] - Enable

Breakpoint control - When you are single-stepping code using Ctrl-W and there is a breakpoint on an instruction, this option tells QDBug if you want to stop, or carry on and execute the instruction.
KEYS : [S] - Stop single emulation
[M] - Meaningless breakpoint (ignore)

Processor type - This tells QDBug whether you wish to emulate an Arm2 or Arm3 processor.
KEYS : [2] - Arm 2
[3] - Arm 3

Load File, PC control - When you load a file which contains an execution address, QDbug can set the PC to this value if this option is enabled.
KEYS : [P] - Set PC to execution address of file (on / off)

Screen options
Use the up / down cursor keys to select the function you wish to change.
QUICK KEY FIND : [SHIFT]+[S] then [S]

Screen mode
The QDBug screen can operate in 9 different screen modes these are :

Mode 1 to 3	-	for normal monitors
Mode 4 to 9	-	for multisync or VGA monitors

The better the monitor you have then the higher resolution screen mode you should chose, select the one that suites you best.
NOTE : Mode 6, this is an Atomwide mode and requires a VIDC enhancer to be fitted.

You can change the screen mode by first pressing [SHIFT]+[F3], which are the main options, then the [S] key for Screen options. Now use the up and down cursor keys so that the arrow is on the 'Mode' option. Now enter your choice by pressing the number keys. Press 'ESCAPE' when your choice is made.

There are other more advanced controls you have over the QDBug screen which are detailed next. All of the following options are available from Options, [SHIFT]+[F3], then Screen, [S]. Next use the up and down cursor keys to chose your option. These are the 'Screen Options'. Press 'ESCAPE' to return to the main control environment.

Mode - This sets the screen mode that QDBug will operate in, as described above.
KEYS : [1] to [9]

Bufferisation - QDBug can save the memory where its screen memory will corrupt. Therefore when you quit QDBug the original memory can be restored. Because you need space in the RMA, Bufferised can only be set if space has been allocated to QDBug. See the *command *QDBug for details on RMA allocation.
KEYS : [A] - Activate, on/off
[B] - Bufferised, on/off (this is set by the computer)

Screen Type - If you are using shadow memory and the shadow option is 'on' then QDBug will use the main screen bank to display its screen.
KEYS : [S] - Shadow, on/off

Program Screen - If the program being debugged uses legal RISC OS calls to write to the VIDC chip then select the RiscOS option. If it writes directly, then select the User option. All of the following options then also have to be set up : Debug screen offset, User screen offset, Colour 0, Colour 1 and Data.
KEYS : [R] - RiscOS
[U] - User

Debug Screen offset - This is only used if 'Program Screen', User mode is selected, and is used to tell QDBug where it can store its screen memory.
If Value < 0 then Value =!(-Value)
0>=Value<=512Kb : physical address
If Value <&2000000 : address in logical RiscOS screen memory.
If Value >=&2000000 : physical address
KEYS : [E] - Enter value

User Screen offset - This is only used if 'Program Screen', User mode is selected, and is used to tell QDBug where the screen memory of the program being debugged will be. The value limits are as detailed above in 'Debug Screen offset'
KEYS : [E] - Enter value

Colour 0 - This is only used if 'Program Screen', User mode is selected, and is used to select colour 0 in the program screen. If you enter a negative value, then this means read colour at address 'value'.
KEYS : [E] - Enter value, range 0 to &FFF

Colour 1 - This is only used if 'Program Screen', User mode is selected, and is used to select colour 1 in the program screen. If you enter a negative value, then this means read colour at address 'value'.
KEYS : [E] - Enter value, range 0 to &FFF

Data - This is only used if 'Program Screen', User mode is selected. The correct VIDC parameters will then have to be entered.
KEYS : [E] - Enter value, use "~" for RiscOS settings.
NOTE : See warning below 'VIDC clock'

VIDC clock - This is only used if 'Program Screen', User mode is selected. It allows you to use a user definable screen frequency.
KEYS : [1] - 24 MHz
[2] - 25.175 MHz
[3] - 36 MHz

WARNING : Please consult VIDC data sheets before changing the 'Data' and 'VIDC clock' settings as some combinations may damage certain types of monitors. Inappropriate settings may also cause the computer to crash.

Changing the Printer Options
If a printout is required of a selected window then some of the following setup's may need adjustment. Press [Ctrl]+[Print] to print a selected window from the main screen.
Use the up / down cursor keys to select the function you wish to change.

QUICK KEY FIND : [Shift]+[F3] then [P]

New line data - Here you must set the sequence of bytes for a 'New line' on the printer.
KEYS : [E] - Enter

Light ON data - This is the sequence of bytes that tell the printer, Light ON. You can enter from one to four bytes.
KEYS : [E] - Enter

Light Off data - This is the sequence of bytes that tell the printer, Light OFF. You can enter from one to four bytes here.
KEYS : [E] - Enter

Driver - The printer driver can be the QDBug Internal Parallel Driver or a RiscOS driver depending on if one is already loaded, and the state of the IRQ. Risc OS printer drivers need the IRQ to be enabled.
KEYS : [R] - Risc OS printer driver, on / off.

IRQ options

This option allows you to change the way IRQ's are handled. Use the up / down cursor keys to select the function you wish to alter.

QUICK KEY FIND : [Shift]+[F3] then [I]

Next Interrupt Handler - The next time QDbug is exited (after a Ctrl-R), the IRQ can be given to RiscOS (in this case QDbug has just use vectors and events). Alternatively the IRQ can be handled by the Debugger allowing it to look directly at keyboard, parallel port and serial port.
This is very useful if you want to trace an IRQ or if the program uses its own IRQ driver.
KEYS : [R] - Risc OS will handle IRQ's
[D] - The debugger will handle IRQ's

Current Interrupt Handler- This informs the user who currently owns the IRQ. Either 'Risc OS' or 'Debugger' will be highlighted.

RiscOS - When in QDBug, if your program uses RiscOS calls to read the keyboard (because for example, you are debugging an IRQ) set the 'Running' option ON, because the state of the keyboard will be held in RiscOS workspace.
KEYS : [R] - Running, toggle on / off

Reset Keyboard on exit - When you leave QDbug (ie by [Ctrl]+[R]) it is advisable that the keyboard is cleared. However some programs including RISC OS itself can't cope with keyboard resets. Use this option with care.

Terminal configuration

QDBug allows the use of a remote terminal. Using the serial port, the terminal can be made to display the QDBug screen leaving the computers screen to show the program being debugged. There is currently support for only one type and that is a VT52 terminal with an 80*25 character screen.

NOTE : The terminals keyboard is null, QDBug is still controlled from the Archimedes.

Use the up / down cursor keys to select the function you wish to change.

QUICK KEY FIND : [Shift]+[F3] then [T]

Type - VT52 only (V0.31).

Baud rate - Can be any Risc OS baud rate.
KEYS : [E] - Edit

Format - There can be 7 or 8 bits data (bit 7 is never used by QDbug). 1 or 2 stop bits, with Parity on or off.
KEYS : [7] - 7 bits data
[8] - 8 bits data
[1] - 1 stop bit
[2] - 2 stop bits
[P] - Parity on / off

Parity - Odd and even parity supported with the option for 1 on TX ignored on RX, or 0 on TX ignored on RX.
KEYS : [O] - Odd parity
[E] - Even parity
[1] - 1 on TX ignored on RX
[0] - 0 on TX ignored on RX

Bottom border - When using a terminal you can switch off the bottom borders so that larger windows can be displayed.

Saving the QDBug configuration
With this option you can save all of your settings to disk. Information on window positions, breakpoints and all other important variables are stored in this configuration file. Once selected you will be prompted for a filename, type one in and presses [Return]. If you call the file 'QDbug_Cfg', QDBug will try and load it automatically from the currently selected directory when it is first loaded, and use it instead of the default settings.

QUICK KEY FIND : [Shift]+[F3] then [A]

When the configuration is saved to disk, it is stored in the form of a text file and this can easily be edited using !Edit. This allows you to create your own selections of windows easily. See Appendix A for the configuration file format.

Load user configuration
This option allows you to load a previously saved configuration file from disk. If the file is called 'QDbug_Cfg' and it is in the same directory, it will be loaded automatically and replace the default settings. Breakpoints, macros, window information and all other user setups are stored in the config. file. This means that you can start where you left off when debugging over several sessions.

QUICK KEY FIND : [Shift]+[F3] then [L]

Functions for execution
The following functions will only work if the 'Running control' is enabled, this is set in the 'Options', 'Execute' menu.

Single stepping
Press [Ctrl]+[W] to single step. The first instruction in the pipeline is executed and the old state of the processor is saved in the 'History'.

QUICK KEY FIND : [Ctrl]+[W]

Running the program
Press [Ctrl]+[R] to leave QDBug and to run the program being debugged. If QDBug was entered from the desktop then this would return you back to the desktop.

QUICK KEY FIND : [Ctrl]+[R]

Set breakpoint and run
This function sets up a breakpoint at PC+4 and runs the program. This is useful if you want to run a BL or SWI. This function will not work in ROM.

QUICK KEY FIND : [Ctrl]+[Q]

Single step, type 2
This is accessed by pressing [Ctrl]+[T].
This is equivalent to [Ctrl]+[Q] if there is a BL as the first instruction in the pipeline. Otherwise it is like [Ctrl]+[W].

QUICK KEY FIND : [Ctrl]+[T]

Skip instruction
This function is accessed by pressing [Ctrl]+[S]. It is equivalent to the single step function, but the condition code of the first instruction in the pipeline is set to 'Never' (NV).

Force instruction
This function is selected by pressing [Ctrl]+[F]. It is equivalent to a single step but its condition code is set to 'Always' (AL).

QUICK KEY FIND : [Ctrl]+[F]

Link R14

This function is accessed by pressing [Ctrl]+[L].

If the first instruction in the pipeline is not a BL, then R14 is saved inside QDBug and R14 is then set to a new memory location inside QDBug. The program is then 'Run'. When the processor uses R14 to return to the caller, QDBug in entered. The status window will display 'Link executed'.

If the first instruction in the pipeline is a BL, then R14 is set to a memory location inside QDBug and the BL is called. QDBug is then entered when the processor returns using R14. This is very useful for when you want to run an entire subroutine that you know is not very interesting. Warning, there will be problems if R14 has been has been saved in the stack.

QUICK KEY FIND : [Ctrl]+[L]

Back in history

This function load the contents of the 'History' back into the registers (R0 to R15). The last entry in the 'History' is then removed. This is useful for back tracking in code.

QUICK KEY FIND : [Ctrl]+[X]

Execute exceptions

If the status window is showing "Undefined Instruction", "Prefetch Abort", "Data Abort", "Address Exception" or if the first instruction in the pipeline is an SWI, then use this function. The SVC is entered and the PC is set to the correct vector (&4 to &14).
This is very convenient if you want to single step your own error routines.

QUICK KEY FIND : [Ctrl]+[E].

Memory functions

QDBug allows various functions to be carried out on bulk memory. You can search, fill and copy using simple commands.

Fill memory with same byte

This 'Fill' function is accessed by pressing [Ctrl]+[F3].
It is entered in the format, 'Start address', 'End address' (or + length) then 'with byte'

QUICK KEY FIND : [Ctrl]+[F3]

Search memory

With this function you can search memory for a byte, word, string or string in a disassembly.

QUICK KEY FIND : [F4]

KEYS : [B] - Byte
[W] - Word
[S] - String
[I] - Instruction.

You cannot search for a string in source loaded into QDBug.

Next match

When a match is found from doing a search use the 'Next Match' to continue. This is done by pressing [Shift]+[F7].

QUICK KEY FIND : [Shift]+[F7]

Previous match

After using 'Next match' you may need to see the previous match. Do this by pressing [Ctrl]+[F7].

QUICK KEY FIND : [Ctrl]+[F7]

Copy memory

This allows you to copy a block of memory from one location to another. It is an intelligent copy so memory can be moved up or down by any amount. It is also done in bytes so that locations do not have to be word aligned. Press [Ctrl]+[F4] to use the 'Copy' function.

QUICK KEY FIND : [Ctrl]+[F4]

Miscellaneous functions

There are a number of extra functions that QDBug offers to help in software debugging.

OS_Exit

This causes QDBug to issue an OS_Exit command to the OS. This means that if a program causes, for instance, a data abort error that you need to clear, you can do so.

QUICK KEY FIND : [Shift]+[F4]

Stop Drive

Sometimes when you jump into QDBug the floppy drive light will remain on. This option allows you to switch the drive off. Just press [Ctrl]+[F2].

QUICK KEY FIND : [Ctrl]+[F2]

Viewing the programs screen

This allows you to view the screen of the program you are debugging. Use [Alt] and cursor keys to change the base address of the screen. All of the debugging controls are still working in this mode but the QDBug screen is not displayed.

QUICK KEY FIND : [V]

View program screen from a start address

To view the program screen from a start address press [Alt]+[V], you will then be prompted for the screen base address. Use [Alt] with the cursor keys to change the base address.

QUICK KEY FIND : [Alt]+[V]

Setting up / down scrolling value

This sets the value in terms of bytes, that [Alt] up or down adds or subtracts when being used.

QUICK KEY FIND : [Shift]+[Alt]+[up]

CLI - Command Line Interpreter

The command line interpreter allows you to type in *commands from QDBug. Everything typed in here is passed onto OS_Cli.

To access this press [F1], the screen should then go blank apart from a '*' in the top right. You can now type in your *command. Press 'Escape' to exit this mode.

QUICK KEY FIND : [F1]

History

Each time a [Ctrl]+W, Q, R is performed, the state of the processor is saved. Press [F5] to see the 'History' of the processor. Use the up / down cursor keys to see more of the data.

QUICK KEY FIND : [F5]

Key press emulation
This option allows you to emulate a key press. To access this feature press [F5].
Once you have pressed [F5] you will asked to select the type of key emulation needed. You have two choices, 'Until' and 'While'.

QUICK KEY FIND : [F5]

KEYS : [U] - Until
[W] - While

NOTE : Other key presses such as [V] are still accessible during an emulation.

Until
The 'Until' choice is selected by pressing [U].
This means your key press will follow the following sequence.
Repeat
execute command
Until <expression>

While
This 'While' choice is selected by pressing [W].
This means your key press will follow the following sequence.
While <expression>
execute command
EndWhile

Syntax
The <expression> can be any expression from Appendix C.
An example would be :
{Until} @=100
{Command} [Ctrl]+[W]
This would cause QDBug to single step for &100 times because @ is an execution counter incremented each time an emulation command is executed.

Stop emulation
To stop the emulation press [Shift]+[Alt]+[Alt]+[Shift]

Calculate expression
To manually calculate an expression, press [F7]. Check Appendix C for the list of valid expressions that you can use.

QUICK KEY FIND : [F7]

Change Register
This option allows you to change the contents of a register. Press [F8] to do this.
Use Appendix C for the list of valid expressions that you can use.
The register can also be PC, PSR, R13_svc, or R8_fiq etc.

QUICK KEY FIND : [F8]

Breakpoints

Overview

The most important feature of any debugger is the ability to add breakpoints. These allow you to put invisible markers on an instruction or memory location and when the processor reaches them QDBug is entered.

It works because QDBug replaces the selected instruction with a 'B qdbug_routine' and when QDBug is entered the breakpoint is replaced by the original instruction. All this is invisible to the program and up to 30 breakpoints can be installed.

No breakpoints can be added to addresses over &2000000 and there can be none in ROM. There can also only be one per memory location.

Breakpoint related keys

Install a breakpoint - [F6]
Add / remove breakpoint- [Ctrl]+[B]
Kill breakpoint - [Shift]+[F6]
Kill all breakpoints - [Shift]+[F8]
List all breakpoints - [Ctrl]+[F8]
Add breakpoint and run - [Ctrl]+[Q]
Run until - [Shift]+[F5]

Installing a breakpoint

There are three possible ways to install a breakpoint into memory. These are with [F6], [Ctrl]+[B] and a QDBug SWI call.

Installing a complex breakpoint, with [F6]

Install a breakpoint by pressing [F6]. You will then be prompted for some breakpoint variables.

Syntax : <address> [, <counter> [, <base> [, <expression>]]]

The first variable is the 'counter'. Breakpoints stop the program if their 'counter' becomes zero.

The second is the 'base'. If the 'base' is >zero when the breakpoint stops the program, the breakpoint is not removed and the counter is reloaded with the 'base' value.

The 'expression' is re-calculated each time the PC executes the breakpoint, if the result is not-zero (ie not false) the 'counter' is decrement by 1, otherwise it is not decremented. If there is no expression, the 'counter' is always decremented (this works like a "-1" expression). See Appendix C for the list of available expressions.

Default parameters are "1,0,-1".This means that the breakpoints stops the program the first time it is executed and is also removed.

Errors : *"Too many breakpoints"* - *You have more than 30 breakpoints*
"Bad address" - *Your address is over &2000000*
"Writing breakpoint error" - *There is no physical memory mapped here*
"Already installed" - *There is already a breakpoint here*

Installing / removing a simple breakpoint, with [Ctrl]+[B]
When using this option to install a breakpoint, you must have a disassembly window selected. The instruction at the top of the window will have the breakpoint attached. If you press [Ctrl]+[B] again it is removed. This function effectively toggles a breakpoint on and off. The breakpoint will have the default parameters, "1, 0, -1".

Run until breakpoint
As a short cut you can 'Run' a program until a particular memory location is reached. Press [Shift]+[F5], next enter the address you are interested in. QDBug will re-enter when this occurs. Again it has the default settings of "1, 0, -1".

QUICK KEY FIND : [Shift]+[F5]

Kill a breakpoint
Press [Shift]+[F6] to kill a breakpoint, you will then be asked for the breakpoint address.

Errors : *"Invalid address"* - *There is no breakpoint at this address*
"Deleting breakpoints error, bad address" - *The MEMC has been re-mapped and the breakpoint cannot be found or deleted*

QUICK KEY FIND : [Shift]+[F6]

Kill all breakpoints
To kill all breakpoints press [Ctrl]+[F6].
This is an intelligent removal so if a breakpoint has been overwritten, the instruction will be left un-altered.

QUICK KEY FIND : [Ctrl]+[F6]

List all breakpoints
To list all the currently defined breakpoints press [Ctrl]+[F8].

QUICK KEY FIND : [Ctrl]+[F8]

Saving and loading breakpoint lists
You can save breakpoint lists for another debugging session by first listing the breakpoints, by pressing [Ctrl]+[F8], then pressing either [F2] or [F3].

KEYS : [F2] - 'Load' previous state
[F3] - 'Save' breakpoint and SWI list
QUICK KEY FIND : [Ctrl]+[F8] then [F2] or [F3]

LABELS

Overview

With the use of QDBug SWI calls in your source code, labels can be used as used in the disassembly of the program. This helps in making the disassembly more readable.
Labels can be register dependant, or absolute. This means that the user can create frames and access them using variables, for example [R12,#Name%].

List Labels

Press [L] to display the labels, this does nothing if there are none loaded.
Once in this mode the following extra functions are available :

Display lists alphabetically	- [A]
Display lists in numeric order	- [N]

Notes :

Redundant labels are not shown in the alphabetic lists. The alphabetic list shows you how a label is interpreted in an expression.
The numeric list shows you how a value is interpreted in a disassembly dump.
If there are labels only in the numeric order list then all labels are redundant.
By default the alphabetic list is selected.

Relocating absolute labels

Press [F8] once in the [L] mode. You will then be asked for the values of the old range and then the value of the start of the new range.

SWI's and labels

Useful SWI calls

"QDbug_RecordLabels"
"QDbug_NoMoreLabels"
"QDbug_NextLabels"

Please refer to Appendix B for more information on the SWI calls that are used when dealing with labels.

Blocks

Overview

QDbug can find the instruction to be executed in a BASIC program, so you can single-step directly inside it. For this to happen, you have to tell QDbug which blocks you are interested in.

For example:

DIM Code% 1024
P%=Code%
[...]

The blocks to record here would be 'Code%'.

Each time an instruction is compiled in the blocks you have defined, QDbug will store the location of instruction, basic slot number, pointer from start of the program, and length of the instruction. This will only work on word-aligned instructions.

For each block defined, QDbug allocates a block of memory of the same length in its workspace. This is the reason why you must define only important blocks.

Blocks can be saved from QDbug's workspace to disk, so you can load it for later use.

For example :

This is very useful if you create a module from a BASIC program. First set up both blocks and labels using the appropriate SWI calls. Next load the blocks, labels and BASIC program into QDBug. You will now be able to single-step the program.

SWI calls for Block functions.

"QDbug_SetBlock"
"QDbug_NewBlock"

See appendix B for full information on these SWI calls.

List Blocks

Press [F9] to list the currently installed blocks. Once in this mode, the following functions then become available.

QUICK KEY FIND : [F9]

Save Blocks

Press [F3] to save blocks and labels for later use.

QUICK KEY FIND : [F9] then [F3]

Load Blocks

Press [F2] to load blocks and labels into QDBug's workspace.

QUICK KEY FIND : [F9] then [F2]

Relocate Blocks
Press [F8] to relocate a block. This allows you to change the address of a block.
For example :
You have compiled a module using BASIC, then saved the block and the labels. You have now loaded your module into the RMA, and wish to debug it. You can load the BASIC source in slot #1, then the blocks and labels. However, you need to set the block to the base address of the module (use F1, *Modules to find this out). Relocating a block will also relocate absolute labels whose value is in the range of the block.
When you press [F8], you will be asked for the number of the block and the new address.

QUICK KEY FIND : [F9] then [F8]

Delete Blocks
Press [F11] to delete a block, you will then be asked for a block number.

QUICK KEY FIND : [F9] then [F11]

Note on blocks and labels
To use the ability to record blocks and labels you need BASIC 1.04 or 1.05 in RAM.
Use *RMFaster BASIC to load it into RAM. Alternatively load BASIC from disk.

Macros

Overview

A macro is a sequence of keys recorded using QDbug. This means that you can create a sequence of key presses and allocate them to a single macro key. Once recorded, macros can be accessed quickly by pressing [Ctrl]+[Alt]+[<macro key>]. They can also be saved and loaded to disk for repeated use.
This is useful if you have a lot of repeated sequences used during debugging.
Macros can also be played by pressing Ctl-Copy.
When you are playing macros using Ctl-Copy for example, what you type in is also executed.

Macros control mode

Press [M] to enter the macros control mode. You will then be offered the following extra functions.

QUICK KEY FIND : [M]

[R] - Record

This allows you to record a macro.
You are asked to press the [Ctrl]+[Alt]+[<macro key>] which will be your macro trigger key.
[<macro key] can be any of the 104 keys in the keyboard except [Break], because [Ctrl]+[Break] will reset the machine.
Next, every key or command you type is recorded into the macro.
To stop recording, press [Shift]+[Alt]+[Alt]+[Shift]. Now press [M] again to tidy the workspace.
The macro is now available by pressing [Ctrl]+[Alt]+[<macro key>].
QDbug then tells you the length of the macro, and the number of bytes lost if the size reserved for the macro was too small.

[S] - Save

This allows you to save some or all of the macros.
You are asked for [S] or [A], which stands for 'Some' or 'All', make your selection, then type in the filename.
If you select [S], you will be asked to press the macro trigger key for the one to be saved. Use [ESC] to end.

[L] - Load

This allows you to load macros from the specified file.

[M] - ReMap

This allows you to change the [<macro key>] of a macro.
You are asked for the current trigger key press, then the new [Ctrl]+[Alt]+[<macro key>].
This is very useful if you have a macro in key [P], for example, and you know that in the file you are about to load also contains a macro in [P].
You can move your macro from [P] to another key before loading the file.

[P] - Play

This is very useful if you want to see what happens when the macro is playing.
You are asked for the macro to play.
Each time you press [Ctrl]+[Copy], another byte or command is read from the macro.
Everything else you type is executed.
To stop playing, you can use [Ctrl]+[BackSpace].

[K] - Kill

You are asked for the macro to kill.
The macro is then cleared from the workspace.

[C] - Clear All

After a confirmation, all the macros in the workspace are cleared.

Set Macro Size

Press [Alt]+[M] to set the amount of workspace (in bytes) that should be allocated to macros.
By default, when you record a macro, all the free memory in the workspace is reserved.
You can set the amount of memory to reserve for the macro with this function.
If you press '0' here the default option is chosen.

QUICK KEY FIND : [Alt]+[M]

SWI handling

Overview

QDbug holds a list of the latest SWI's executed by the operating system.
There is also another list which contains only SWI's called from RAM. This is more useful because it contains only calls made by user programs.
You can also interrupt the execution of up to 16 SWI's. This is useful, for example, to single-step your own SWI's, or to 'Run' a program until its access a RAM resident module via an SWI.

SWI control mode

Press [S] to enter the SWI control mode. You will then be offered the following extra functions :

QUICK KEY FIND : [S]

[H] - History

This shows you the latest SWI's used by the system.
It is a list which records SWI's called from RAM only, this is because it is often more interesting for debugging purposes.

[L] - List

This shows you the SWI numbers that will be intercepted by QDbug.
Each SWI is numbered.
There can be up to 16 SWIs installed.

[A] - Add

Add a SWI to the list, ie you want to intercept another SWI.
When a SWI is called and it is in the list, QDBug is entering and the text "SWI intercepted" is displayed in the Status window.
If all the 16 slots are full, this option is not shown. You must remove one first.

[R] - Remove

This allows you to remove a SWI from the list, when you do this the SWI will not be intercepted again.
You are asked for the number of the SWI slot (1-16) not the SWI number.

[C] - Clear

This allows you to clear all of the installed SWI's. You are first asked for confirmation, then they are cleared.
No more SWIs will be intercepted.

Appendix A - **configuration file format**

The following text describes the QDBug configuration file format.

#d_SWI is followed by "H" (hex), "I" (Internal table) or "C" (call RiscOS).
#d_Stacks is followed by the registers (0...9,A...F).
#d_Format is followed by "D" (decimal), "H" (hex) or "B" (both).
#d_DInfo is followed by "Y" (yes) or "N" (no).
#d_MType is followed by "M" (Motorola) or "I" (Intel).
#d_BInfo is followed by "Y" (yes) or "N" (no), "2" to "7", "P" (punctual) or "S" (standing).
#d_Tab is followed by "1" to "9".
#e_SWI is followed by "E" (emulate) or "F" (follow)
#e_Run is followed by "Y" (yes) or "N" (no).
#e_EStop is followed by "M" (meaningless) or "S" (stop).
#e_ARM is followed by "2" (Arm2) or "3" (Arm3).
#e_Load is followed by "N" (do nothing) or "P" (set PC).
#s_Mode is followed by the mode number ("1" to "9").
#s_Buffer is followed by "Y" (yes) or "N" (no).
#s_Shadow is followed by "Y" (yes) or "N" (no).
#s_SType is followed by "R" (RiscOS) or "U" (user).
#s_DebOff is followed by the value of Debug Screen Offset in hex.
#s_UsrOff is followed by the value of User Screen Offset in hex.
#s_Color0 is followed by the value of Colour0 in hex.
#s_Color1 is followed by the value of Colour1 in hex.
#s_Data is followed by "#" and the list of the registers.
#s_Clock is followed by "1" (24Mhz), "2" (25.175Mhz) or "3" (36Mhz).
#p_NewLin is followed by the bytes.
#p_LighON is followed by the bytes.
#p_LigOFF is followed by the bytes.
#i_Handle is followed by "R" (RiscOS) or "D" (debugger).
#k_RiscOS is followed by "Y" (yes) or "N" (no).
#k_Reset is followed by "Y" (yes) or "N" (no).
#t_Type is followed by "V" (VT52).
#t_Baud is followed by the hex value used by RiscOS.
#t_Format is followed by the hex value used by the RiscOS in two digits.
#t_Border is followed by "B" (bottom line) or "-" (no bottom line).
#t_On is followed by "Y" (yes) or "N" (no).

Windows coordinates are saved for each of the modes.

#w_Mode sets the mode of the following windows.
Mode 0 is the terminal.
#w_XYWH is followed by the number of the window and by ":" (if open) or "" (if closed)

and by the four digit decimal value of X, Y, Width and Height.

#w_Font is followed by the font height (1 or 2) to use for each window.

#w_Type is the type of each windows.
1 is disassembly
2 is memory dump
3 is source
4 is pipeline
5 is BASIC

#w_Addr is followed by the start address of each window (2 to 7).

#w_Lock is followed by the number of the window and ":" and the lock string.

Appendix B - SWI calls

The following text describes the SWI calls that QDBug supports. A higher level of debugging can be achieved when using these calls.

The SWI Chunk prefix is 'QDbug_' and the base number is &44B80.

QDbug_SetBreakPoint (&44B80)
On entry:
R0=address of the instruction

Set a default breakpoint at the memory location given.

QDbug_KillBreakPoint (&44B81)
On entry:
R0=address of the breakpoint

This removes the breakpoint.

QDbug_StartRecording (&44B82)
On entry:
R0=key code of the macro (0-&67, see page 122 of the RiscOS Programmer's Reference Manual)
This is the same as if the user had chosen option *[R]ecord* from the *[M]acros* command.

QDbug_StopRecording (&44B83)
On exit:
R0=bytes in the macro
R1=bytes lost
This is the same as if the user had pressed *[M]* inside QDbug after stopping the macro.

QDbug_PlayMacro (&44B84)
On entry:
R0=key code of the macro
The next time QDbug is entered the macro will be played.

QDbug_Break (&44B85)
QDbug sets a breakpoint on the next instruction, so the program is halted.
(This SWI is used by the command *Break)

QDbug_DebugScreen (&44B86)
On entry:
R0=screen address or -1 to read
On exit:
R0=current screen address
This is used to set the "Debug Screen" value (as in the options, see chapter options).

QDbug_UserScreen (&44B87)
On entry:
R0=screen address or -1 to read
On exit:
R0=current screen address
This is used to set the "User Screen" value (as in the options, see chapter options).

QDbug_Colour0(&44B88)
On entry:
R0=colour (0-&FFF) or -1 to read
On exit:
R0=current colour 0
This is used to set the "Colour0" value (as in the options, see chapter options).

QDbug_Colour1 (&44B89)
On entry:
R0=colour (0-&FFF) or -1 to read
On exit:
R0=current colour 1
This is used to set the "Colour1" value (as in the options, see chapter options).

QDbug_VIDCClock (&44B90)
On entry:

R0=new VIDC Clock value or -1

On exit:

R0=current VIDC Clock value

This is used to set the "VIDC Clock" value (as in the options, see chapter options).
Value is:

#0 for 24Mhz
#1 for 25.175Mhz
#2 for 36Mhz

QDbug_ScreenData (&44B91)
On exit:

R0=screen data pointer

This is a pointer on a 14 word lists for 14 of the VIDC registers.
These registers are used to set the new mode when the "User" screen is selected.
You can read or write into these registers.
But the following order must be used:

word 0 - &80xxxxxx (HCR)
word 1 - &84xxxxxx (HSWR)
word 2 - &88xxxxxx (HBSR)
word 3 - &8Cxxxxxx (HDSR)
word 4 - &90xxxxxx (HDER)
word 5 - &94xxxxxx (HBER)
word 6 - &9Cxxxxxx (HIR)
word 7 - &A0xxxxxx (VCR)
word 8 - &A4xxxxxx (VSWR)
word 9 - &A8xxxxxx (VBSR)
word 10 - &ACxxxxxx (VDSR)
word 11 - &B0xxxxxx (VDER)
word 12 - &B4xxxxxx (VBER)
word 13 - &E0xxxxxx (CR)

When a RiscOS screen is selected, these values are read from the address (R0+14*4).
You too can read this list to find out the RiscOS screen data values.

QDbug_SetBlock (&44B92)
On entry:

R0=address of the block
R1=size of the block (in bytes)

This clears all blocks in workspace and looks for BASIC programs present (so use it after loading libraries).
A block of memory of the same size is created in QDbug's workspace and every instruction assembled inside the block defined by R0 and R1 will be stored inside the workspace (format: 5 bits for basic slot number, 7 bits for instruction length, 20 bits for offset of the instruction in the source: inside 1MegaByte).

QDbug_NewBlock (&44B93)
On entry:

R0=address of the block
R1=size of the block (in bytes)

This creates another block of the same size in QDbug's workspace, and instructions assembled inside the block defined by R0 and R1 will be stored inside.
You can have as many blocks as memory permits.

QDbug_RecordLabels (&44B94)
On entry:

R0=maximum number of labels
R1=size of the names

This clears the labels present in QDbug, and create a block in workspace ready to receive R0 labels and another block of R1 bytes ready to receive the names of the labels.
A label is recorded when basic sets a variable preceded by "." to P%.

QDbug_NoMoreLabels (&44B95)
On exit:

R0=0 or error pointer if V set
R1=number of labels found
R2=total size of the names (in bytes)

This stops the recording of labels and creates two lists of labels, one sorted numerically and the other sorted alphabetically.
This SWI uses the SWI "OS_HeapSort" to sort the labels.
If several labels have the same name, they are all removed from the alphabetic list.

QDbug_NextLabels (&44B96)
On entry:

R0=frame pointer
R1=0 to store P%, non-zero to store O%

Use this SWI to tell QDbug if the next labels are absolute (R0=15) or register-relative (R0=0 to 14, usually 12 for R12), and which value is to be stored (P% or O%?).
You have to call this SWI after a QDbug_RecordLabels because this SWI doesn't change these options, so they are undefined.

Appendix C - List of expressions

This appendix contains the list of the expressions that QDBug will understand.

Operators :

+ - / *

AND OR ORR EOR XOR BIC

<< >> >>>

= < > <= >= <>

! ?

Numbers :

[n] = address of the window n, where n is from 2 to 7

[`x,x,x...] transforms the "x" in bytes of PSR type :

x= N, Z, C, V, I, F, USR, USER, SVC, IRQ, FIRQ, FIQ

Any number is HEX by default

The prefix \ means a decimal number and % binary number

@ is the number of iterations made by function [F5] (Emulation)

R0 to R15 represent the registers

PC, PSR are derived from R15 in the following way : PC = R15 BIC &FC000003
: PSR = R15 AND &FC000003

The suffixes _user, _usr, _irq, _firq, _fiq and _svc can be used to access particular registers.

One to four characters between "" are converted into ascii.

More than 4 characters between "" are sent to the SWI OS_SWINumberFromString and the value returned is used.

Appendix D - Key controls

This appendix contains the list of QDBug control keys and their respective functions.

Function key controls

KEYS	*FUNCTION*
F1	- CLI
Shift F1	- Current Directory
Ctrl F1	- Set Current Directory
F2	- Load File
Shift F2	- Load Source
Ctrl F2	- Stop Drive
F3	- Save File
Shift F3	- 'OPTIONS' menu
Ctrl F3	- Fill
F4	- Search
Shift F4	- OS_Exit
Ctrl F4	- Copy
F5	- Emulation
Shift F5	- Run Until
Ctrl F5	- History
F6	- Set a Breakpoint
Shift F6	- Kill a Breakpoint
Ctrl F6	- Info
F7	- Calculate Expression
Shift F7	- Next Match
Ctrl F7	- Previous Match
F8	- Change Register
Shift F8	- Kill all Breakpoints
Ctrl F8	- Breakpoint List
F9	- Blocks
Shift F9	- Delete Window
Ctrl F9	- Move / Stretch windows
F10	- Swap character height
Shift F10	- Change Window Type
Ctrl F10	- Motorola Memory dump
Ctrl Shift F10	- Change Window, Text / BASIC
F11	- Edit
Shift F11	- Clear screen
Ctrl F11	- Intel Memory Dump
F12	- Execute
Shift F12	- Set Window Start Address Locker
Ctrl F12	- Set Start Address

General controls

KEYS	*FUNCTION*
Shift Shift Alt	- Enter QDBug
Ctrl Print	- Print Window
Tab	- Select Next Window
Shift Tab	- Select Previous Window
L	- 'LABEL' mode
S	- 'SWI' mode
F	- Free
V	- View program screen
M	- 'MACRO' mode
Alt V	- View program screen from address
Alt M	- Set Macro Size
Shift Alt up	- Set Value for up / down Scrolling
Alt 1 to Alt 7	- Select specific window
Cursor keys	- Move within selected windows
Home	- Open Current Window to full Screen
Escape	- Return to main control screen
Copy	- Cancels the effect of 'Home'
Ctrl B	- Set Default Breakpoint

"Running control" functions

KEYS	*FUNCTION*
Ctrl W	- Single Step
Ctrl R	- Run
Ctrl Q	- Set Breakpoint and Run
Ctrl T	- Single Step, Type 2
Ctrl S	- Skip Instruction
Ctrl F	- Force Instruction
Ctrl L	- Link R14
Ctrl X	- Back in History
Ctrl E	- Execute Exceptions

Appendix E - QDBug Quick Reference Card

Pull out this card and use it for easy access to QDBug's features.

General controls

KEYS	*FUNCTION*
[Shift]+[Shift]+[Alt]	- Enter QDBug
[Ctrl]+[Print]	- Print Window
[Tab]	- Select Next Window
[Shift]+[Tab]	- Select Previous Window
[L]	- 'LABEL' mode
[S]	- 'SWI' mode
[F]	- Free
[V]	- View program screen
[M]	- 'MACRO' mode
[Alt]+[V]	- View program screen from address
[Alt]+[M]	- Set Macro Size
[Shift]+[Alt]+[up]	- Set Value for up / down Scrolling
[Alt]+[1] to [Alt]+[7]	- Select specific window
Cursor keys	- Move within selected windows
[Home]	- Open Current Window to full Screen
[Escape]	- Return to main control screen
[Copy]	- Cancels the effect of 'Home'
[Ctrl]+[B]	- Set Default Breakpoint

"Running control" functions

KEYS	*FUNCTION*
Ctrl W	- Single Step
Ctrl R	- Run
Ctrl Q	- Set Breakpoint and Run
Ctrl T	- Single Step, Type 2
Ctrl S	- Skip Instruction
Ctrl F	- Force Instruction
Ctrl L	- Link R14
Ctrl X	- Back in History
Ctrl E	- Execute Exceptions

QDBug Quick Reference Card - side 2

Expressions

Operators :

+ - / *

AND OR ORR EOR XOR BIC

<< >> >>>

= < > <= >= <>

! ?

Numbers :

[n] = address of the window n, where n is from 2 to 7

[`x,x,x...] transforms the "x" in bytes of PSR type :

x= N, Z, C, V, I, F, USR, USER, SVC, IRQ, FIRQ, FIQ

Any number is HEX by default

The prefix \ means a decimal number and % binary number

@ is the number of iterations made by function [F5] (Emulation)

R0 to R15 represent the registers

PC, PSR are derived from R15 in the following way : PC = R15 BIC &FC000003
: PSR = R15 AND &FC000003

The suffixes _user, _usr, _irq, _firq, _fiq and _svc can be used to access particular registers.

One to four characters between "" are converted into ascii.

More than 4 characters between "" are sent to the SWI OS_SWINumberFromString and the value returned is used.

Appendix E - QDBug Quick Reference manual copy

This is a copy of the pull out Quick Reference Guide for your convenience.

General controls

KEYS	*FUNCTION*
[Shift]+[Shift]+[Alt]	- Enter QDBug
[Ctrl]+[Print]	- Print Window
[Tab]	- Select Next Window
[Shift]+[Tab]	- Select Previous Window
[L]	- 'LABEL' mode
[S]	- 'SWI' mode
[F]	- Free
[V]	- View program screen
[M]	- 'MACRO' mode
[Alt]+[V]	- View program screen from address
[Alt]+[M]	- Set Macro Size
[Shift]+[Alt]+[up]	- Set Value for up / down Scrolling
[Alt]+[1] to [Alt]+[7]	- Select specific window
Cursor keys	- Move within selected windows
[Home]	- Open Current Window to full Screen
[Escape]	- Return to main control screen
[Copy]	- Cancels the effect of 'Home'
[Ctrl]+[B]	- Set Default Breakpoint

"Running control" functions

KEYS	*FUNCTION*
Ctrl W	- Single Step
Ctrl R	- Run
Ctrl Q	- Set Breakpoint and Run
Ctrl T	- Single Step, Type 2
Ctrl S	- Skip Instruction
Ctrl F	- Force Instruction
Ctrl L	- Link R14
Ctrl X	- Back in History
Ctrl E	- Execute Exceptions

QDBug Quick Reference Card (manual copy) - side 2

Expressions

Operators :
+ - / *
AND OR ORR EOR XOR BIC
<< >> >>>
= < > <= >= <>
! ?

Numbers :
[n] = address of the window n, where n is from 2 to 7
[`x,x,x...] transforms the "x" in bytes of PSR type :
x= N, Z, C, V, I, F, USR, USER, SVC, IRQ, FIRQ, FIQ

Any number is HEX by default

The prefix \ means a decimal number and % binary number

@ is the number of iterations made by function [F5] (Emulation)

R0 to R15 represent the registers

PC, PSR are derived from R15 in the following way : PC = R15 BIC &FC000003
: PSR = R15 AND &FC000003

The suffixes _user, _usr, _irq, _firq, _fiq and _svc can be used to access particular registers.

One to four characters between "" are converted into ascii.

More than 4 characters between "" are sent to the SWI OS_SWINumberFromString and the value returned is used.