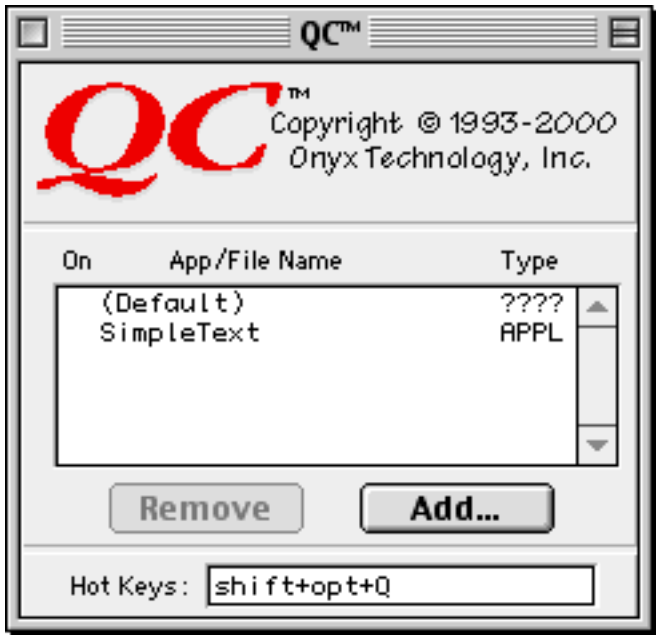
# QuickStart

**QC is** a development tool for stress testing application and stand alone code during runtime. It is designed to be thorough, fast, and easy to use. **QC requires** System 7 in order to function. This document is intended to give you a quick overview of the QC Control Panel and the items in it. The descriptions here are based on the current QC 1.5 release. Previous 68k versions (i.e. 1.1.3) differ where noted. The documentation that ships with QC contains more detailed explanations of all the tests.

The **main Control Panel** window lets you target code to be tested and set the hot key for on the fly testing. Individual test options are available for each application targeted for testing.

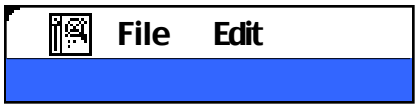



Clicking on the **Add** button brings up a standard file dialog that allows you to select applications, cdevs and other code resources to be added to the **target list**. These new **target** programs will have the same testing options as the **default** target.

The **Remove** button will remove the currently selected target from the list (the default target cannot be removed). You cannot add or delete items from the list when testing is active.

Clicking in the **hot key field** allows you to replace the current hot key setting with the key combination of your choice (command key combinations are not allowed).

**Pressing the hot key** at any time will cause QC to begin testing the front-most application when the key down event is processed (WaitNextEvent). If an application being tested did not previously exist in the list it will be added and the default settings will be used.

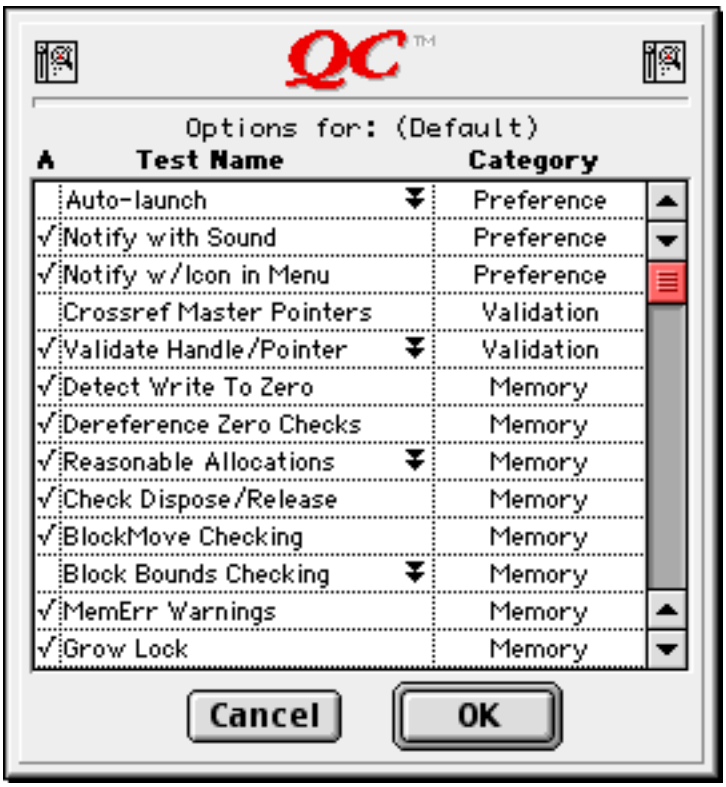



While QC is running a small icon will flash over the Apple menu and your system will slow down (how much depends on the test settings). There are notification options that allow you to specify whether QC will beep when activated/deactivated and/or whether the small icon is rotated in the Apple menu. Pressing the hot key again will immediately stop QC.

QC 1.5 runs on all PowerPC Macintosh systems using System 7 or later. VM, 32-bit addressing, and Modern Memory are all supported. Balloon help is available for all window and dialog items.

**Pricing** for QC is $99 plus postage and handling (typically $5 domestic US, varies for international and courier deliveries). Five and ten pack pricing available. Site licenses are available for 50 copies or more. Contact Onyx Technology for further information. Internet: 'support@onyxtech.com'. Tele: (941) 795-7801. Fax: (941) 795-5901.

Double clicking on a target item (or default) in the main control window brings up the **test list dialog** for that item. This dialog shows a scrolling list of tests that QC can perform, and available preference settings. Clicking on the first column of any line item will cause that test/preference to be **on** whenever QC is testing the application. Changes to default settings apply to activation on an application that is not already in the main Control Panel list. For example, if you activate on TeachText using the hot keys and it was not already in the launch list, default settings will be used.




**Auto-launch** will turn QC on automatically whenever the program is launched (the *double triangle* symbol indicates additional options are available by double-clicking).

**Notify with Sound** will cause QC to emit an audible sound when testing is activated and deactivated. With this option off, no sound is made. **NOTE** that previous to QC 1.2, this test item was named 'Notify with Beep'.

**Notify w/Icon in Menu** will cause QC to rotate a small QC icon in the Apple menu when testing is active. With this option off, no icon is displayed.

**Crossref Master Pointers** will verify that every relocatable handle is pointed to by a master pointer within a non-relocatable block.

**Validate Handle/Ptr** verifies Handles and Pointers as they are passed to memory manager traps.

**Detect Write To Zero** will check on each trap call to see if location zero has been overwritten (as would happen when writing to a purged handle, for example). The PC of the trap where this was detected will be reported.

**Dereference Zero Checks** places a value in location zero that will cause a bus error if dereferenced by the target.

**Reasonable Allocations** will report any memory allocations that exceed a user determined size (negative or otherwise erroneous values passed to NewHandle, for instance).

**Check Dispose/Release** will warn you if you are calling DisposeHandle on a resource or ReleaseResource on a handle.

**BlockMove Checking** will verify that the destination address range for BlockMove calls is within a single allocated block of memory (if the destination is within the tested heap - BlockMove's to the stack and low memory locations are not tested).

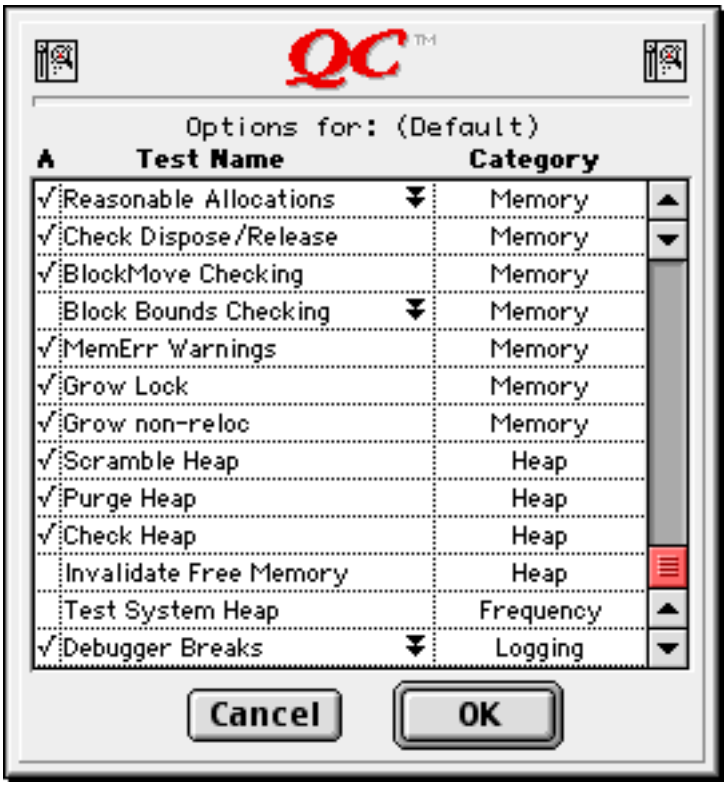
**Block Bounds Checking** will detect writes past the end of a block for all blocks in the tested heap(s).

**MemErr Warnings** will break into the debugger every time a MemErr is detected after a memory manager call. This test can be useful in finding code that does not handle failed memory operations but can also be annoying as QC reports anytime MemErr is set. You can then determine if your app handles the possibility of a MemErr at that point. **NOTE** that previous to QC 1.2, this test item was named 'MemErr Detection'.

**Grow Lock** will break into the debugger every time an attempt is made to grow a locked handle.

**Grow non-Reloc** will break into the debugger every time an attempt is made to grow non-relocatable block.

**Scramble Heap** will cause all relocatable blocks to be moved whenever it is possible for them to move (exposing faulty de-referencing assumptions).




**Purge Heap** causes purgeable handles to be purged whenever memory allocation takes place.

**Check Heap** will verify the heap's structure prior to performing any memory manager calls that allocate memory.

**Invalidate Free Memory** fills all non-allocated heap memory with values that will cause a bus error if dereferenced by the target program.

**Test System Heap** will cause QC to perform all tests on the system heap zone. If this item is unchecked QC will limit testing to the application heap zone.

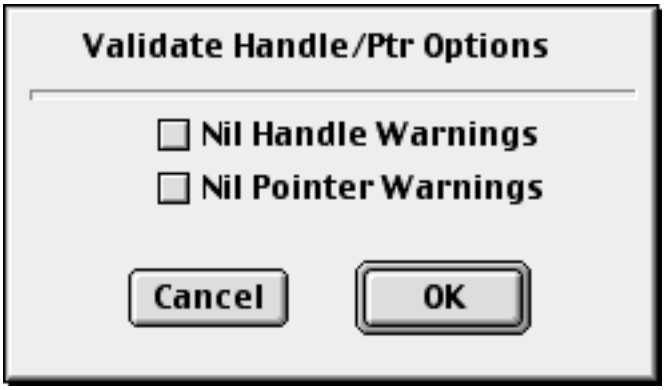
**Debugger Breaks**, if on, will cause QC to report the errors detected via a DebugStr trap. If off, QC will emit a short beep sound whenever a bug is detected.

Quick tips includes option+clicking on a test to toggle all tests on/off, and command+clicking on a test to toggle all tests of the same category on/off.

Double clicking on the AutoLaunch test name will bring up the **AutoLaunch Options Dialog.** Use the radio buttons to configure when QC will begin automatically testing the target code.

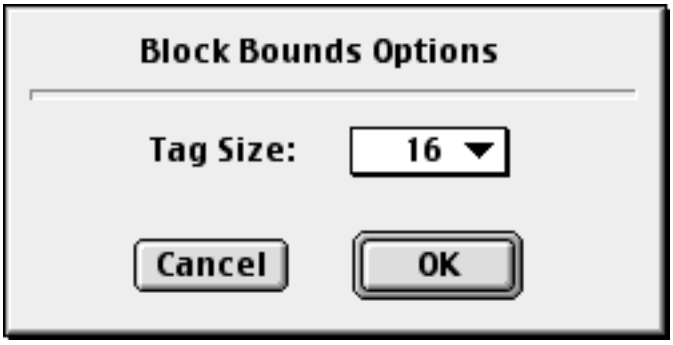

You can set QC to begin testing when the target application calls _InitGraf or have QC test the heap when a code resource is loaded by selecting the **Use Get1Resource** option. This will engage QC testing when Get1Resource is called within the named target code with the parameters specified by **type** and **id**. Testing will automatically deactivate when that resource is disposed of. (For more extensive control see the QC manual for information on the Application Programming interface).

Double clicking on the Validate Handle/Ptr test name will bring up the **Validate Handle/Ptr Options** dialog. Use this dialog to control whether QC is to report the usage of a NIL Handle and/or NIL Pointer in Memory Manager calls. **NOTE** that this options dialog only exists in QC 1.2 or later.




Double clicking on the Reasonable Allocation test name will bring up the **Reasonable Allocation Options** dialog**.** Use this to define what is meant by an "unreasonable" memory allocation. This is useful for identifying calls that are way out of line and may cause a crash later.


Double clicking on the Block Bounds Checking test name will bring up the **Block Bounds Options** dialog. Use this to define the size of the tag added to heap blocks to detect block overwrites. Increasing the number will increase chances of catching overwrites that possibly write that far past the end of blocks.



Double clicking on the Debugger Breaks test name will bring up the **Debugger Break Options** dialog. Depending on the low and/or high level debugger you are using, you might want to specify to QC how it reports errors to you. For example, on a PowerMac using SysBreakStr is the only way for some high level debuggers to intercept and display a user break to you. Other debuggers prefer DebugStr instead. **NOTE** that this options dialog only exists in QC 1.2 or later.


For programmers who want complete control over how QC is testing their code, the **QC Application Programming Interface (QCAPI)** is available. Using the API, you can control all test settings that are provided from the Control Panel interface, directly from your code. There are additional abilities the API supplies; like the ability to save the current testing state, change test settings for a block or area of code, then restore the saved state for the remainder of the application.

QC also ships with a sample application called **BadAPPL** which illustrates many of the types of errors QC catches. The application's source code is also supplied as an example of how to use the QCAPI to control testing from within your code.

QCAPI libraries are available for Metrowerks and MPW development environments.

# Debugging with CodeWarrior™

If you are a Metrowerks CodeWarrior user, you can use QC to more easily debug applications that you are stress testing. As of CodeWarrior 6 (May, 1995), the CodeWarrior debuggers are "QC-Aware". This means that when testing with QC, the debuggers now catch QC detected errors and display them to you along with the source code of where the error was detected. This greatly enhances the debugging ability of QC detected errors because you not only have complete stack crawls for your code, variable inspection, but you also have the source causing the error available at your fingertips.

The version of Metrowerks Debuggers that support QC are version 1.2 or later. QC versions 1.1.3 or later are needed to provide the information the debuggers expect.

For more information about CodeWarrior, please contact Metrowerks, Inc.

**Canada and International**
Metrowerks Inc.
1500 du College, suite 300
St. Laurent, QC
H4L 5G6 Canada
voice: (514) 747-5999
fax: (514) 747-2822

**U.S.A.**
Metrowerks Corporation
Suite 310
The MCC Building
3925 West Braker Lane
Austin, TX 78759-5321
voice: 512-305-0400
fax: 512-305-0440

**Metrowerks Mail Order**
voice: (800) 377-5416 or (419) 281-1802
fax: (419) 281-6883

World Wide Web site (Internet): http://www.metrowerks.com
Registration information (Internet): register@metrowerks.com
Technical support (Internet): support@metrowerks.com
Sales, marketing, & licensing (Internet): sales@metrowerks.com
AppleLink: METROWERKS
America OnLine: goto: METROWERKS
Compuserve: goto: METROWERKS
eWorld: goto: METROWERKS

CodeWarrior is a trademark of Metrowerks, Inc.