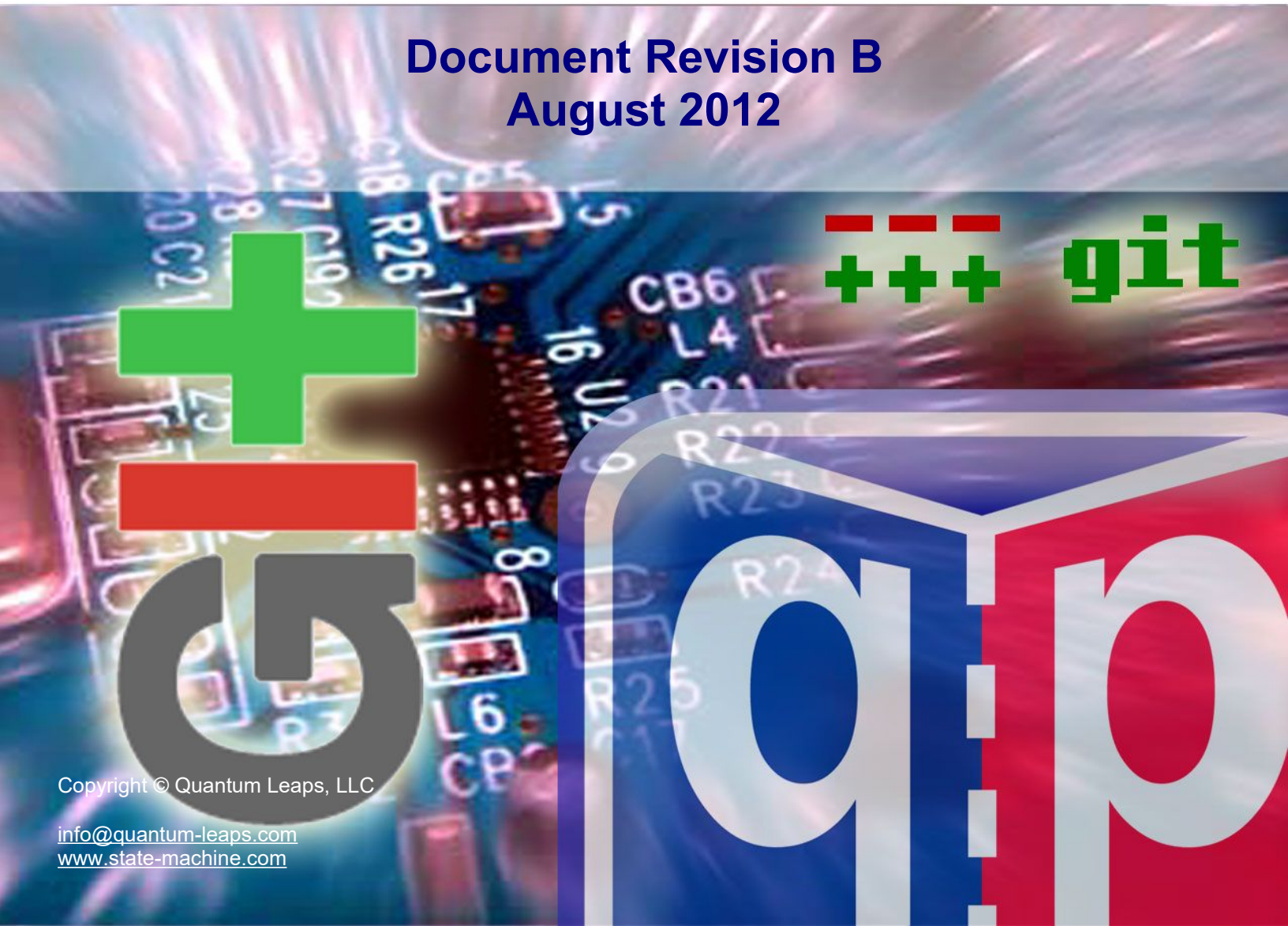
Quantum® LeaPs

innovating embedded systems

Application Note

Accessing QP™ Git Repositories on SourceForge.net

Document Revision B
August 2012



info@quantum-leaps.com
www.state-machine.com

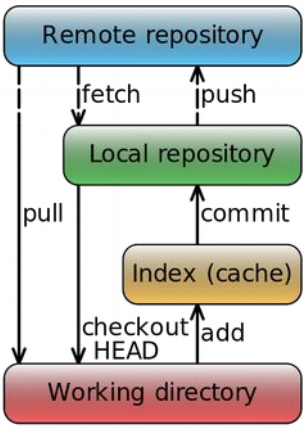
Table of Contents

1 Introduction

This Application Note describes how to access the QP™ source code via Git repository on **SourceForge.net**. Specifically, you will learn how to install Git on Windows and how to clone the complete source trees for QP/C, QP/C++, QP-nano and Qtools. You will also learn how to fetch updates to the source trees as they are pushed to the QP Git repository.

1.1 About Git

Git is a free & open source, Distributed Source Code Management (DSCM), a tool for software developers which supports collaborative development of software within a team, and the tracking of changes to software source code over time.

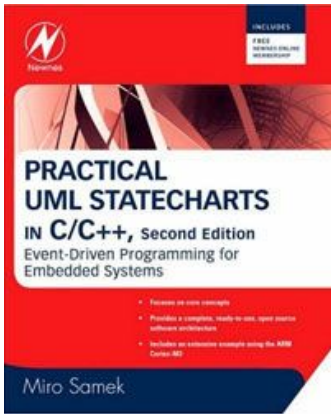


Git is used by developers, and advanced users who need the very latest changes to the software (before releases occur). Software users generally do not need Git; typically they will download official file releases made available by the project instead. Developers should familiarize themselves with Git by reading the **Git documentation**.

One of the many advantages of Git over other Source Code Management systems, such as CVS or SVN is that Git keeps the whole history in just one directory (.git) and does **not** pollute the source tree with the history information.

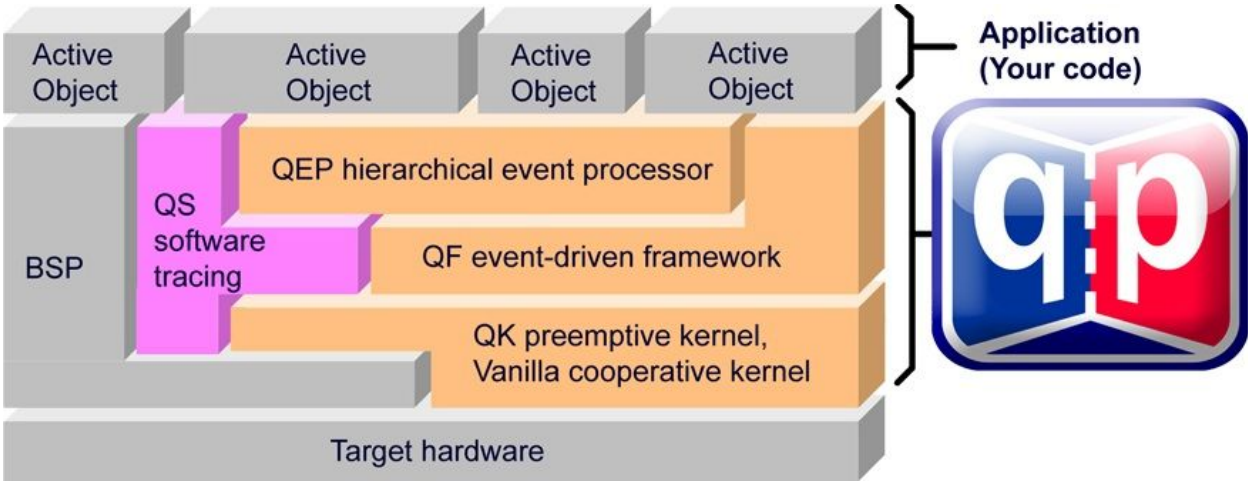
1.2 About QP™

QP™ is a family of very lightweight, open source, state machine-based frameworks for developing event-driven applications. QP enables building well-structured embedded applications as a set of concurrently executing hierarchical state machines (UML statecharts) directly in C or C++. QP is described in great detail in the book "*Practical UML Statecharts in C/C++, Second Edition: Event-Driven Programming for Embedded Systems*" [PSiCC2] (Newnes, 2008).



As shown in Figure 1, QP consists of a universal UML-compliant event processor (QEP), a portable real-time framework (QF), a tiny run-to-completion kernel (QK), and software tracing instrumentation (QS). Current versions of QP include: **QP/C**™ and **QP/C++**™, which require about 4KB of code and a few hundred bytes of RAM, and the ultra-lightweight **QP-nano**, which requires only 1-2KB of code and just several bytes of RAM.

Figure 1: QP components and their relationship with the target hardware, board support package (BSP), and the application

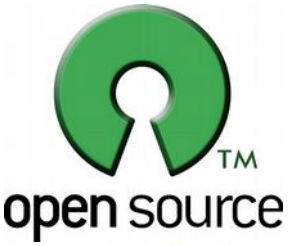


QP can work with or without a traditional RTOS or OS. In the simplest configuration, QP can completely **replace** a traditional RTOS. QP includes a simple non-preemptive scheduler and a fully preemptive kernel (QK). QK is smaller and faster than most traditional preemptive kernels or RTOS, yet offers fully deterministic, preemptive execution of embedded applications. QP can manage up to 63 concurrently executing tasks structured as state machines (called active objects in UML).

QP/C and QP/C++ can also work with a traditional OS/RTOS to take advantage of existing device drivers, communication stacks, and other middleware. QP has been ported to POSIX (Linux/Embedded Linux, QNX, INTEGRITY), Win32 (Windows/Windows Embedded, WindowsCE), VxWorks, ThreadX, uC/OS-II, and other popular OS/RTOS.

1.3 Licensing QP

The **Generally Available (GA)** distributions of QP available for download from the www.state-machine.com/downloads website are offered under the same licensing options as the QP baseline code. These available licenses are:



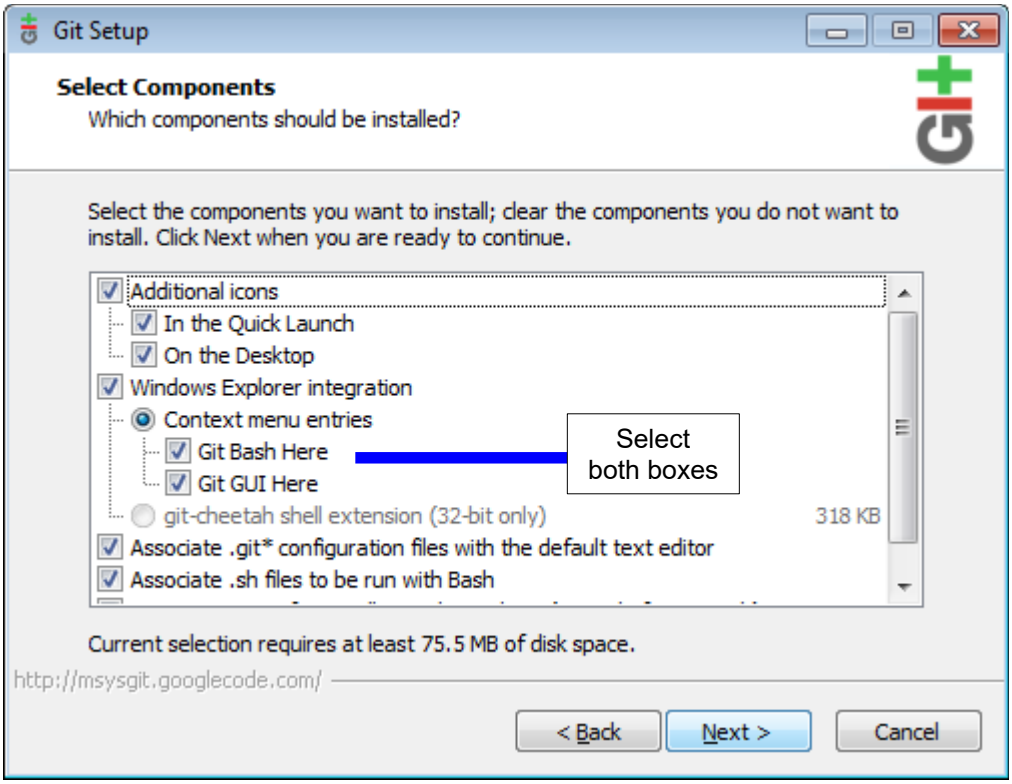
- The GNU General Public License version 2 (GPL) as published by the Free Software Foundation and appearing in the file `GPL.TXT` included in the packaging of every Quantum Leaps software distribution. The GPL *open source* license allows you to use the software at no charge under the condition that if you redistribute the original software or applications derived from it, the complete source code for your application must be also available under the conditions of the GPL (GPL Section 2[b]).

- One of several Quantum Leaps commercial licenses, which are designed for customers who wish to retain the proprietary status of their code and therefore cannot use the GNU General Public License. The customers who license Quantum Leaps software under the commercial licenses do not use the software under the GPL and therefore are not subject to any of its terms.

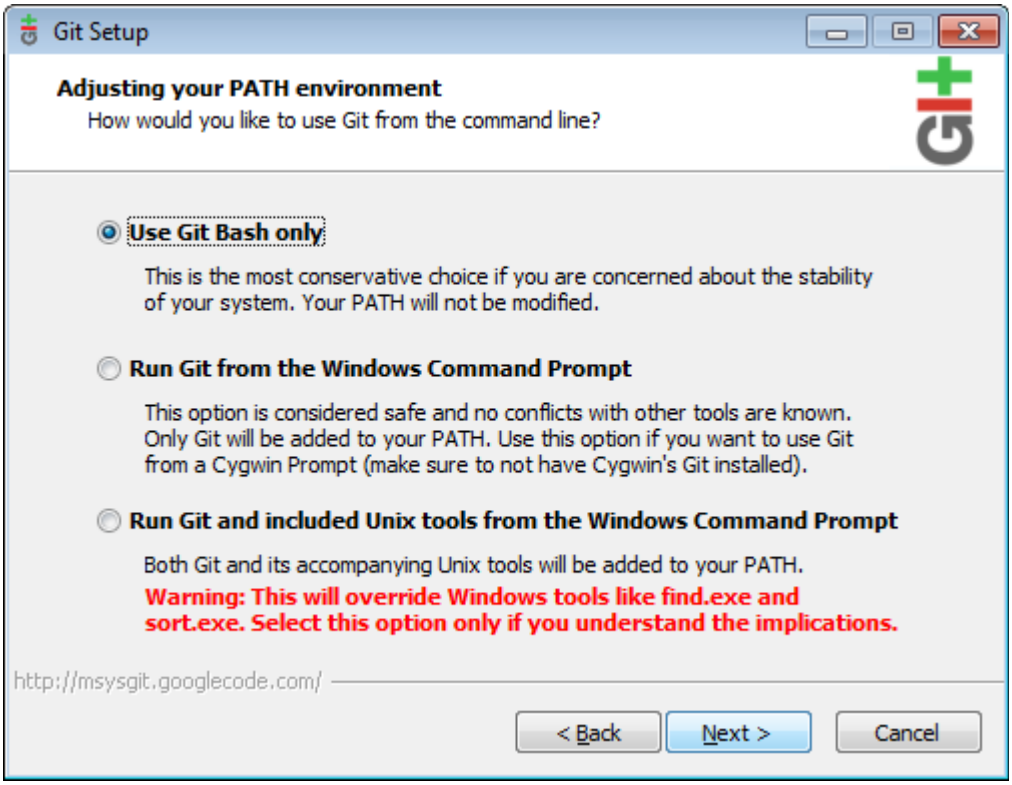
For more information, please visit the licensing section of our website at: www.state-machine.com/licensing.

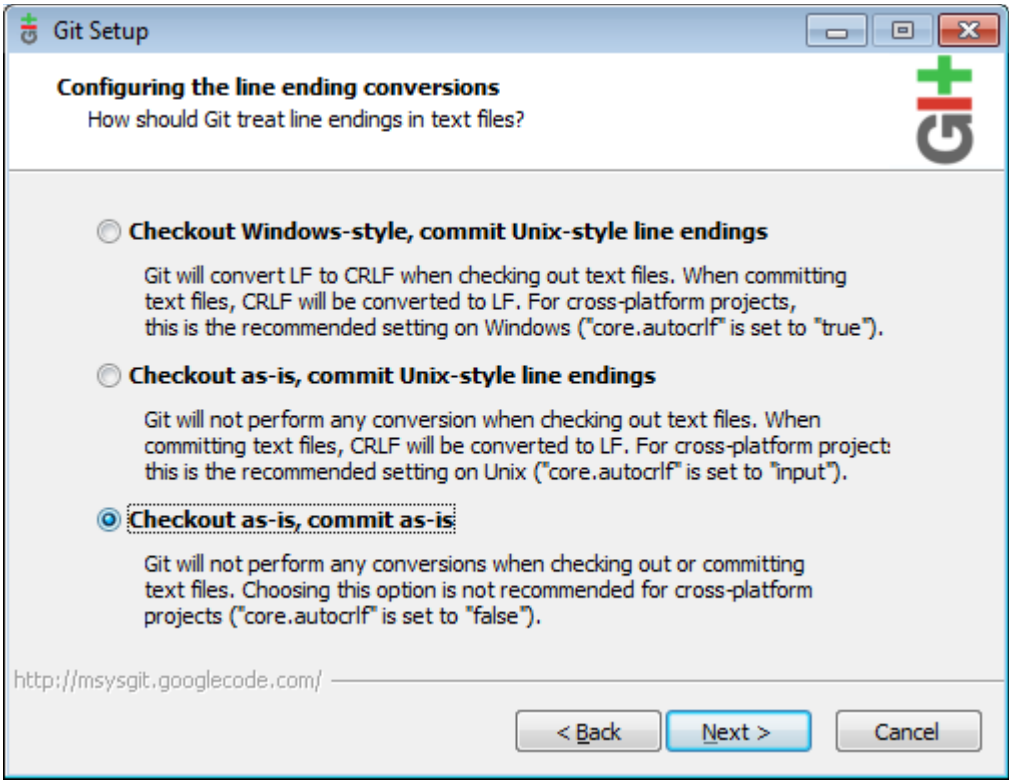
2 Installing Git on Windows

In order to access QP source tree on SoucreForge.net, you need to download and install the Git software. For Windows hosts, you can the free **msysgit**, which is available from http://code.google.com/p/msysgit/-downloads/list. This download is a single executable (e.g., `Git-1.7.9-preview20120201.exe`) which installs the entire Git system. While going through the installer, you will want to check the options to add Windows Explorer integration when you right click on a folder. The following screen shots show the recommended installation choices.



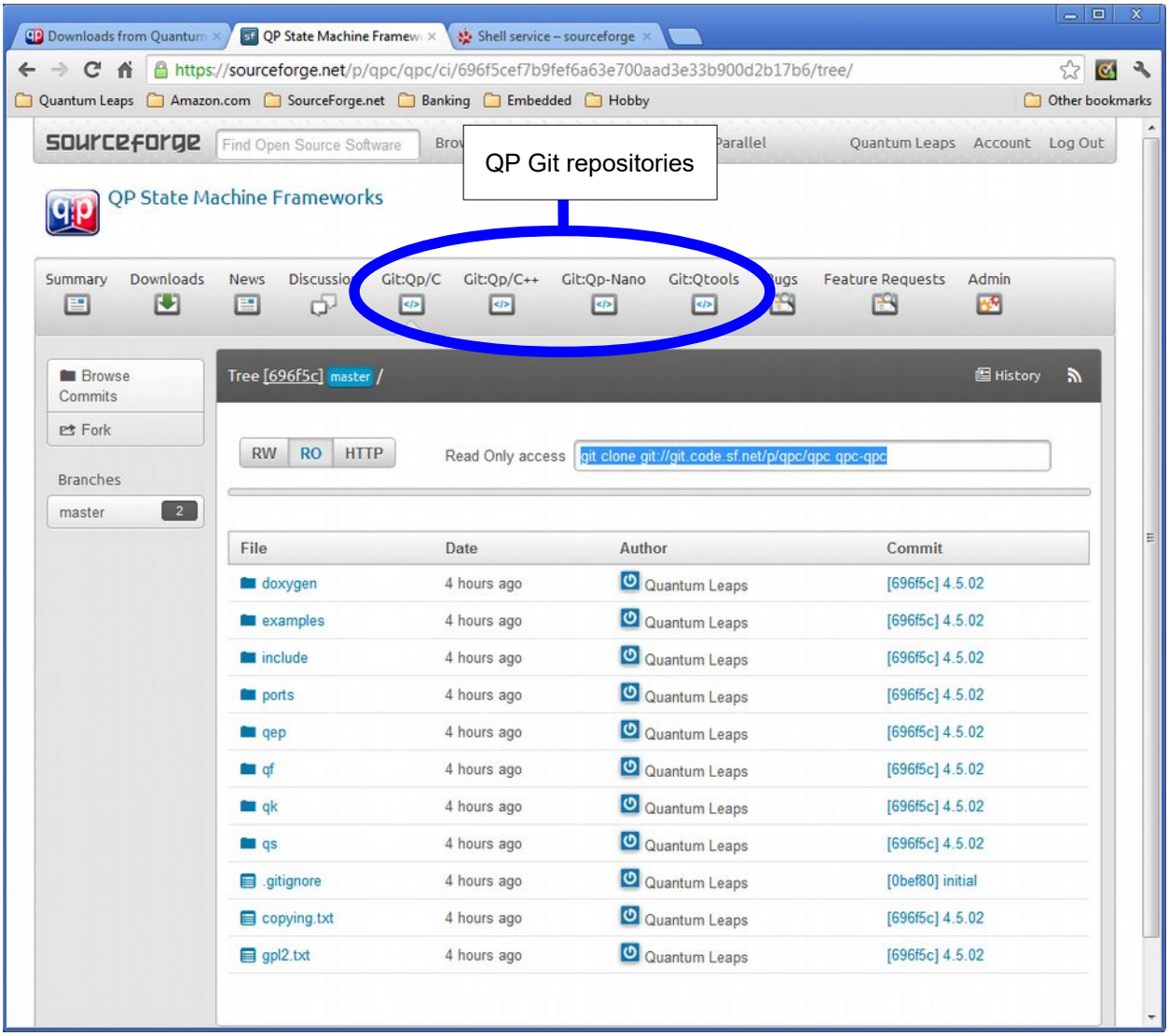
NOTE for Configuring the Git line ending conversions: The QP source code is already written in the Unix (LF) end-line convention (see the Quantum Leaps Coding Standard [QP-Code]), so no additional conversions are necessary. Therefore, the recommended setting for the QP frameworks is "Checkout as-is, commit as-is", which is also the safest policy for binary files.





3 QP Git Repositories on SourceForge.net

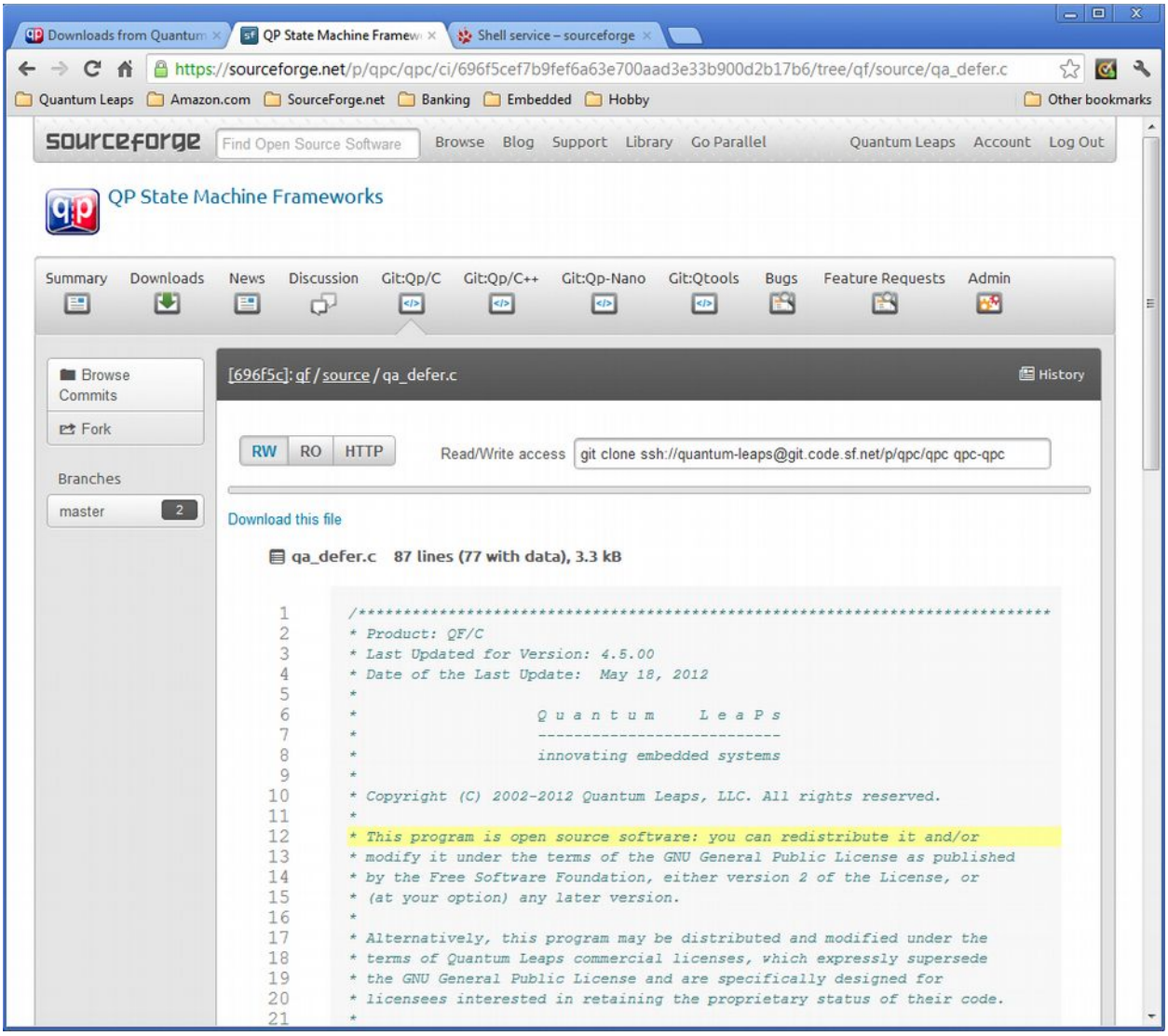
Each QP framework type (QP/C, QP/C++, and QP-nano) as well as Qtools are stored in a separate SourceForge.net Git repository. You can view the QP Git repositories ("repos" in Git lingo) by clicking on the Git::xx menu items. For example, the screen shot below shows the content of the Git:Qp/C repo:



3.1 Exploring QP Source Tree Directly on SourceForge.net

SourceForge.net provides a web interface to the Git repos, so that you can explore the source code trees directly in your web browser. (The upcoming sections describe how to copy the entire source code to your local disk.)

For example, to explore the Git:Qp/C source tree (QP/C framework), you click on the folder icon, for example qf, next source, and finally on the file you want to view, for example `qa_defer.c`. This opens the following web page:



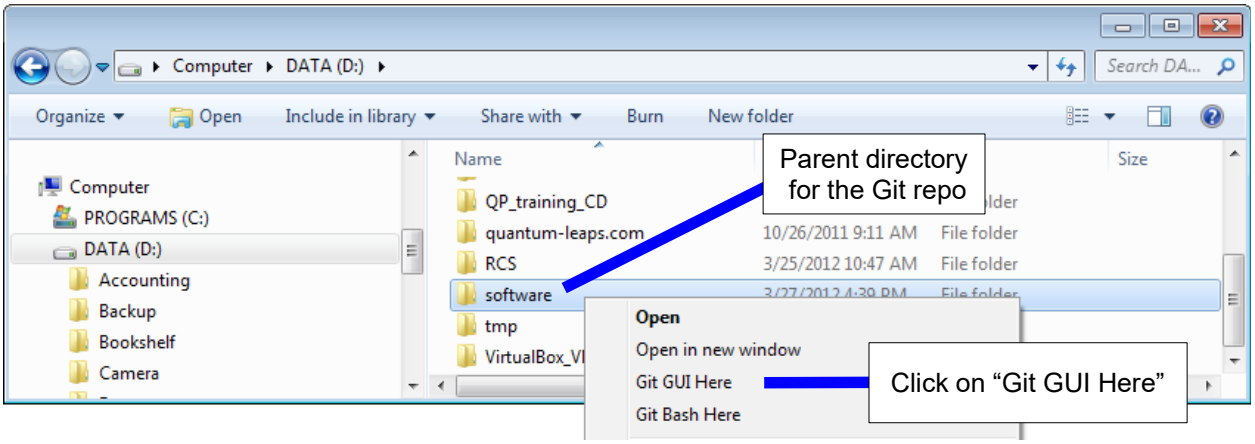
To go back to the root of the tree, you click on the **master** button in the right pane.

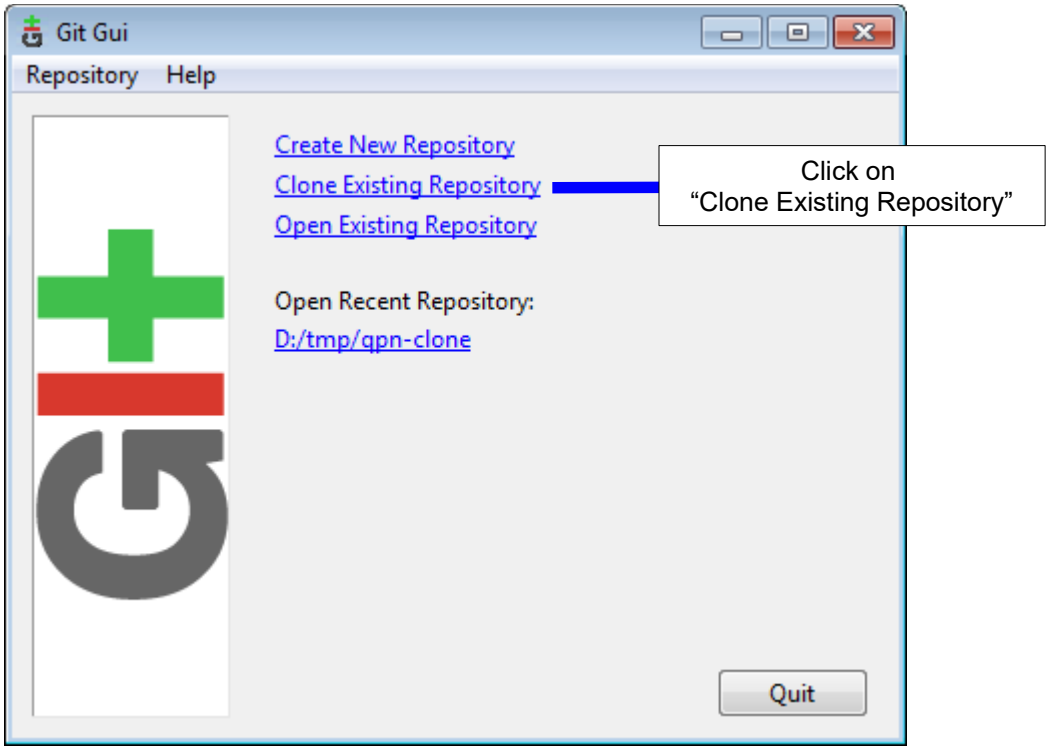
4 Cloning the QP Git Repos from SourceForge.net

To access any of the QP Git repos for the first time, you need to **clone** the Git repo. Cloning is the process of copying entire repo from one location (SourceForge.net) to another (your local directory) and therefore it can take a bit of time. However, to get any subsequent updates, you only need to **fetch** the latest changes, which is much faster.

4.1 Cloning with GitGUI

To clone a Git repo with GitGui, first open the Windows Explorer and point it to the directory where you want to locate the repository. Next, you **right-click** on the directory and select "Git GUI Here":

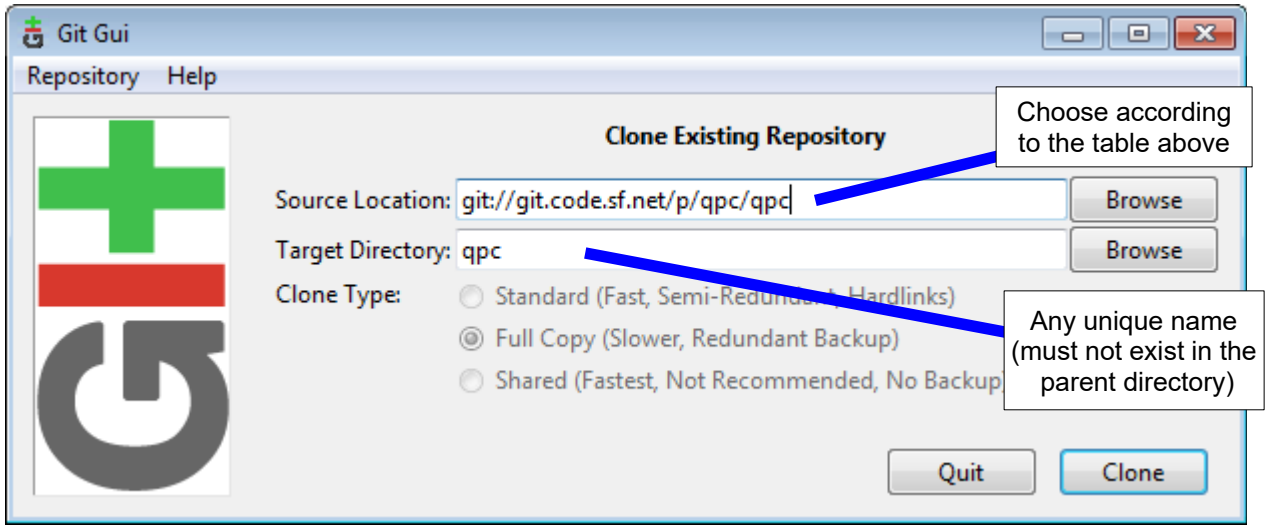


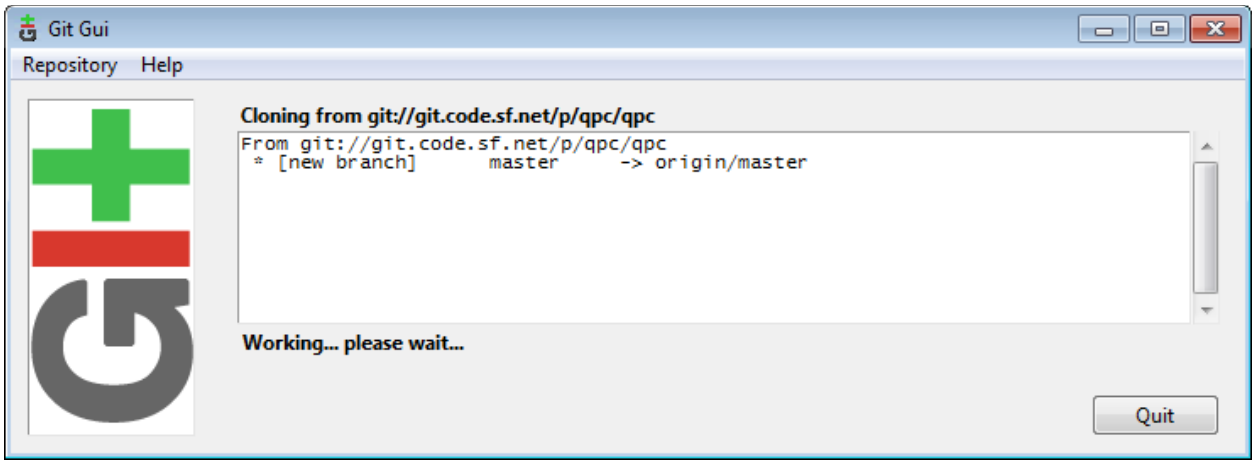


In the next dialog box "Clone Existing Repository", Git asks you about the Source Location of the Git repo and the Target Directory, where you want to clone it to. The following table contains the Source Locations and recommended Target Directory names for the QP repositories. You can use different names for Target Directory names.

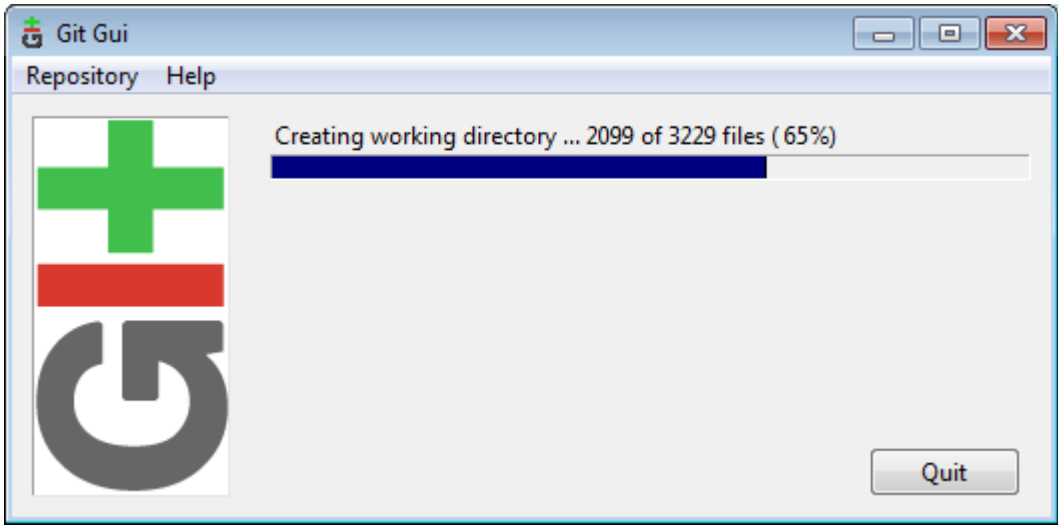
QP repo	Source Location	Target Directory
QP/C	`git://git.code.sf.net/p/qpc/qpc`	`qpc`
QP/C++	`git://git.code.sf.net/p/qpc/qpcpp`	`qpcpp`
QP-nano	`git://git.code.sf.net/p/qpc/qpn`	`qpn`
Qtools	`git://git.code.sf.net/p/qpc/qtools`	`qtools`

NOTE: The specified Target Directory should **not** exist in the parent directory you selected in the Windows Explorer to host the repo.

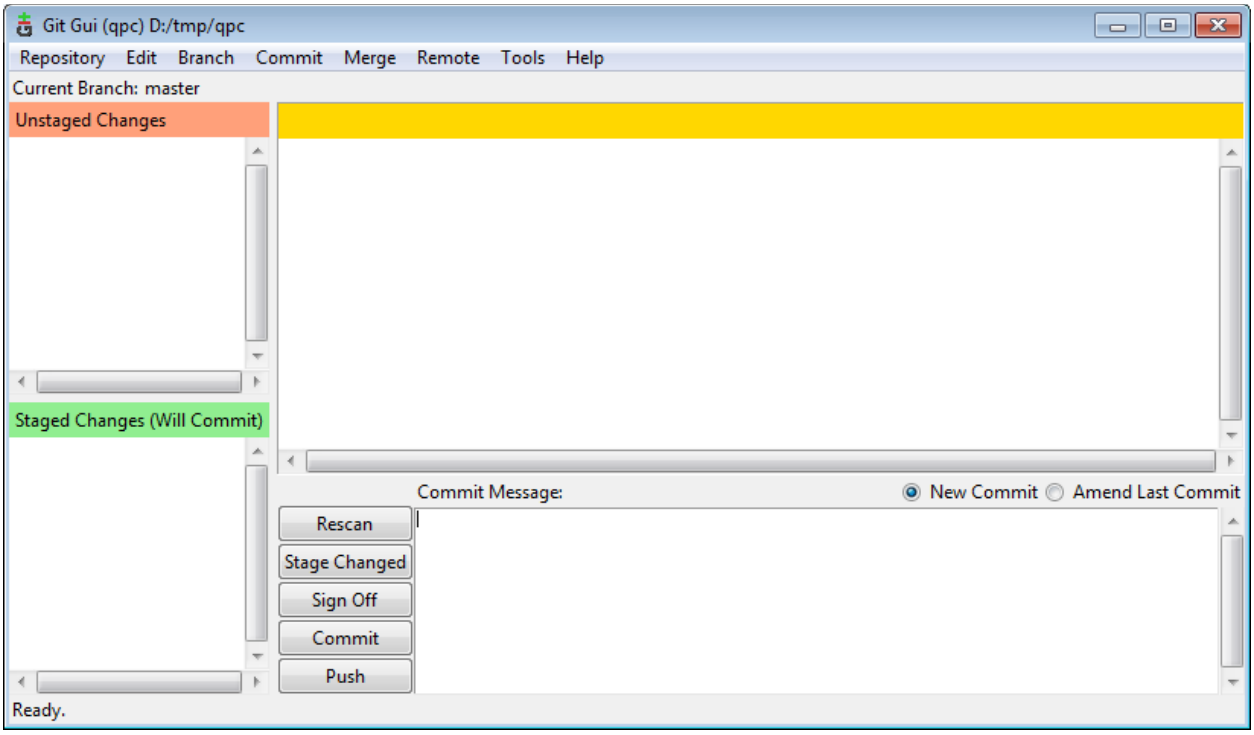




Depending on your network speed, the cloning can take several seconds, during which time Git shows you the progress bar "Creating working directory...".



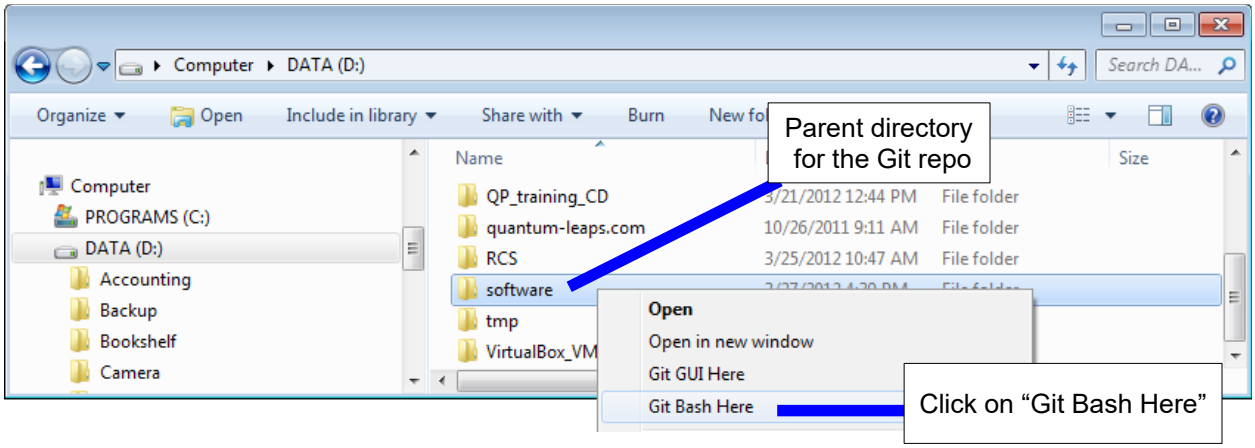
Finally, the cloning process completes and the GitGUI user interface opens up. The QP repo is copied to your local disk and you can explore it in the local directory on the disk.



NOTE: To learn more about using GitGUI, you can read "An Illustrated Guide to Git on Windows" at http://nathanj.github.com/gitguide/tour.html

4.2 Cloning with Git Bash

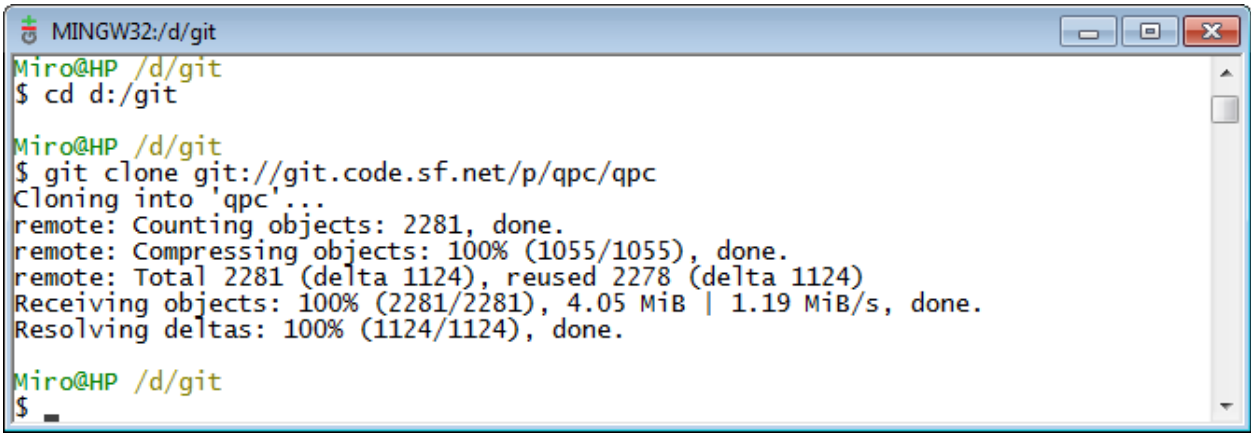
To clone a Git repo with Git Bash, first open the Windows Explorer and point it to the directory where you want to locate the repository. Next, you **right-click** on the directory and select "Git Bash Here":



Alternatively, you can open the Git Bash application and type "`cd <parent folder>`" at the command prompt. Either way, you will end up with Git Bash ready to clone the QP repo.

To clone the `qpc` repo you type the following command:

```
$ git clone git://qpc.git.sourceforge.net/gitroot/qpc/qpc qpc
```



The following table lists the Source Locations and recommended directory names for all other QP Git repos on SourceForge.net:

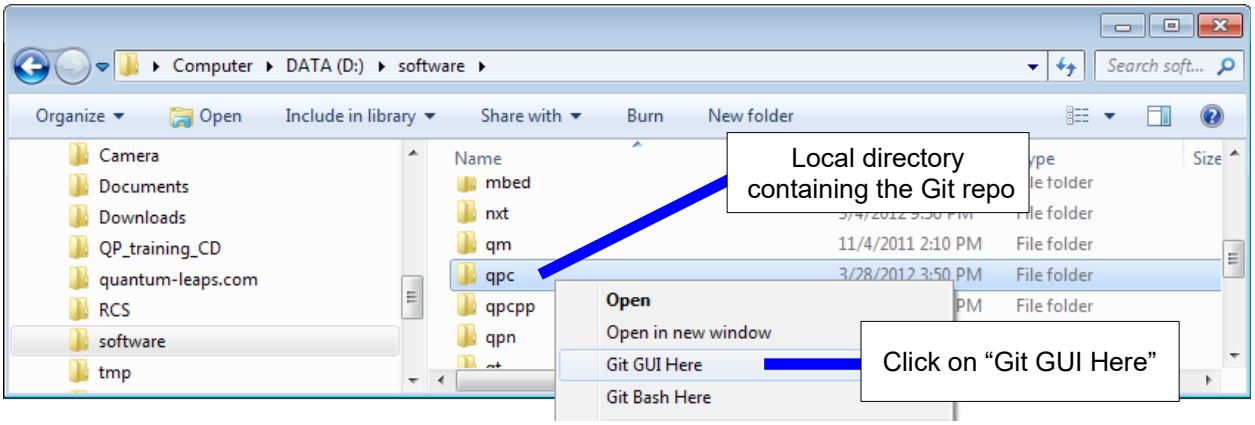
QP repo	Source Location	Target Directory
QP/C	`git://git.code.sf.net/p/qpc/qpc`	`qpc`
QP/C++	`git://git.code.sf.net/p/qpc/qpcpp`	`qpcpp`
QP-nano	`git://git.code.sf.net/p/qpc/qpn`	`qpn`
Qtools	`git://git.code.sf.net/p/qpc`	`qtools`

5 Fetching (Updating) the QP Git Repos from SourceForge.net

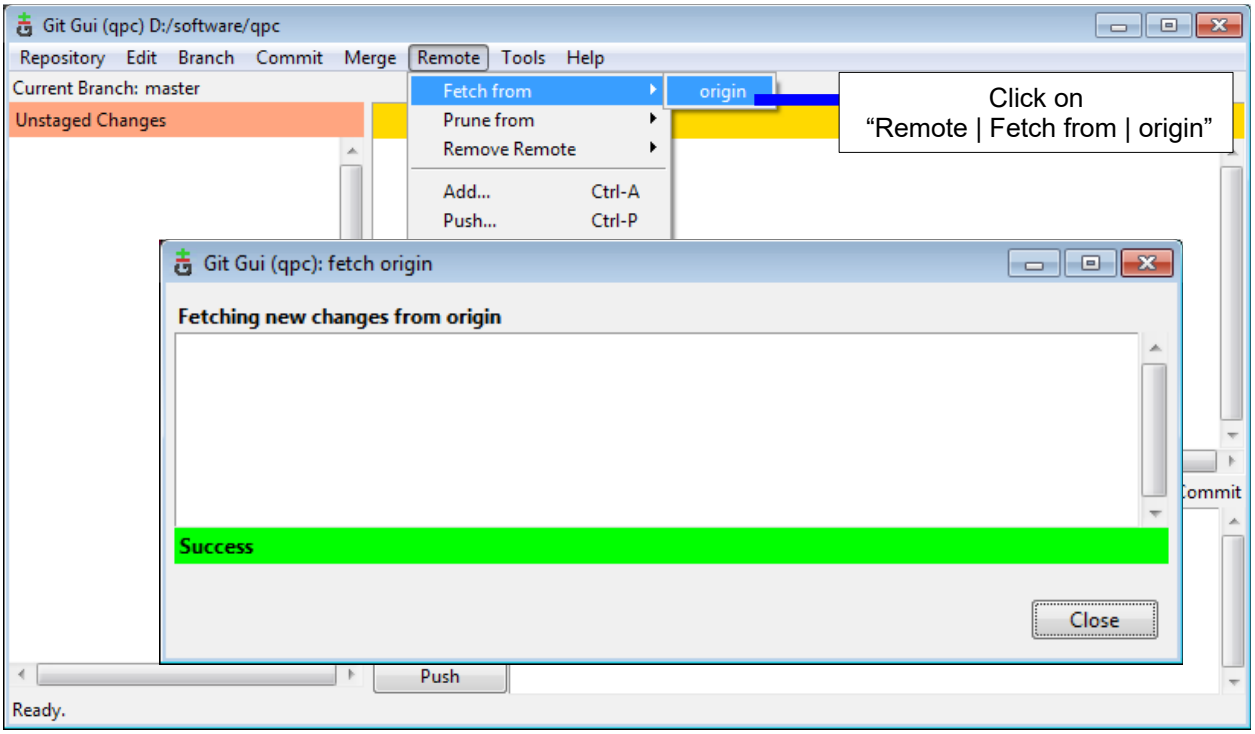
Once you have a cloned QP repo on your local drive you only need to fetch the latest changes to keep it up-to-date. Fetching a Git repo is quick and easy.

5.1 Fetching with GitGUI

To fetch a Git repo with GitGui, first open the Windows Explorer and point it to the directory where you want to locate the repository. Next, you **right-click** on the directory and select "Git GUI Here":



Once GitGui opens up you select the menu **Remote | Fetch from | origin** option.

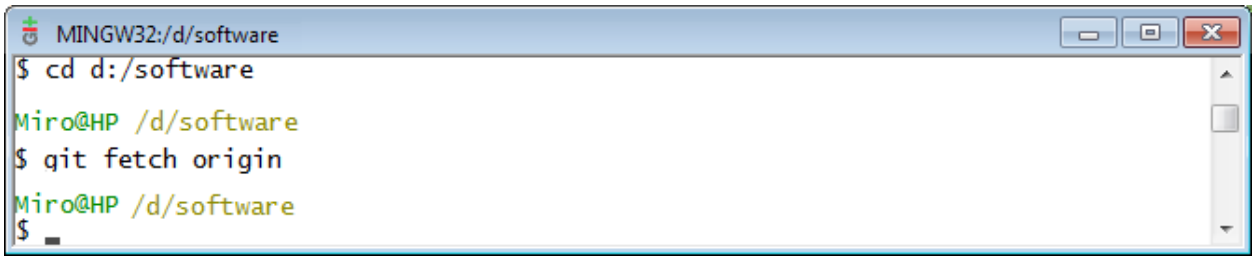


5.2 Fetching with Git Bash

To fetch a Git repo with Git Bash, you can either open Git Bash in the repo folder directly from the Windows Explorer, as described before, or you can launch the Git Bash application and type "`cd <repo folder>`" at the command prompt. Either way, you will end up with Git Bash ready to work with the repo.

To fetch the repo you type the following command:

```
$ git fetch origin
```



6 Summary

The complete source trees of all QP framework types and the Qtools collection are available as Git repositories (repos) on SourceForge.net. You can very easily access the source code with Git and keep it up to date with very fast updates. You can also keep track of all changes in the repository as well as your own local changes, which you can easily merge with the master branch of the code.

This AppNote described only the very basic operations of cloning and fetching the QP Git repos from SourceForge.net. Please refer to the extensive Git documentation for more information about using Git.

7 Related Documents and References

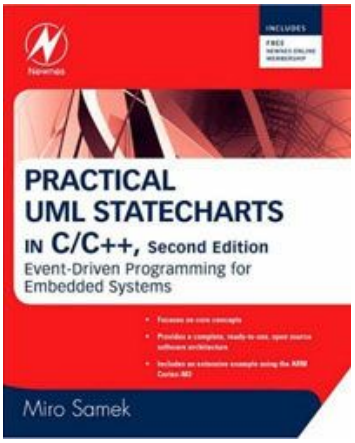
Document	Location
[Git] Main Git website	http://git-scm.com/
[MsysGit] Git for Windows, free downlaod	http://code.google.com/p/msysgit/downloads/list
[GitGui] An Illustrated Guide to Git for Windows	http://nathanj.github.com/gitguide/tour.html
[Git-SourceForge] SourceForge.net documentation for Git users	http://sourceforge.net/apps/trac/sourceforge/wiki/Git
[Git CheatSheet] Git cheat sheet	https://github.com/AlexZeitler/gitcheatsheet
[QL-Code 11] "Application Note: C/C++ Coding Standard", Quantum Leaps, LLC, 2011	http://www.state-machine.com/resources/AN_QL_Coding_Standard.pdf

8 Contact Information

Quantum Leaps, LLC
103 Cobble Ridge Drive
Chapel Hill, NC 27516
USA

+1 866 450 LEAP (toll free, USA only)
+1 919 869-2998 (FAX)

e-mail: info@quantum-leaps.com
WEB : http://www.quantum-leaps.com
http://www.state-machine.com



"*Practical UML Statecharts in C/C++, Second Edition: Event Driven Programming for Embedded Systems*", by Miro Samek, Newnes, 2008