QNX® Software Development Platform

Welcome to the QNX Software Development Platform

For Windows®, Linux®, and QNX® Neutrino® hosts

Contents

List of Figures

About This Guide

Welcome to the QNX Software Development Platform is intended to introduce you to QNX SDP and the QNX Momentics Tool Suite, and help you start developing applications for the QNX Neutrino RTOS.
For information about installing QNX SDP, see the *Installation Guide* or the installation notes in the Download area of our website.

The following table may help you quickly find information in this guide:

For information about:	**See this chapter:**
An overview of QNX SDP, other QNX products, target systems, and licensing	What is the QNX Software Development Platform?
the QNX Momentics Tool Suite	Getting Started
Getting started with the documentation, technical support	How to Get Help
Terms used in QNX documentation	Glossary

Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications. The following table summarizes our conventions:

Reference	**Example**
Code examples	`if( stream == NULL )`
Command options	**`-lR`**
Commands	**`make`**
Environment variables	**PATH**
File and pathnames	**`/dev/null`**
Function names	*exit()*
Keyboard chords	Ctrl-Alt-Delete
Keyboard input	**`something you type`**
Keyboard keys	Enter
Program output	`login:`
Programming constants	NULL
Programming data types	`unsigned short`

continued…

Reference	Example
Programming literals	`0xFF`, `"message string"`
Variable names	*stdin*
User-interface components	**Cancel**

We use an arrow (→) in directions for accessing menu items, like this:

> You'll find the **Other...** menu item under **Perspective→Show View**.

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

CAUTION: Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

WARNING: Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Note to Windows users

In our documentation, we use a forward slash (`/`) as a delimiter in *all* pathnames, including those pointing to Windows files.

We also generally follow POSIX/UNIX filesystem conventions.

Chapter 1

What is the QNX Software Development Platform?

In this chapter. . .

Welcome to the QNX Software Development Platform

Thank you for choosing the QNX Software Development Platform. It includes everything you need to build and maintain your QNX Neutrino-based embedded system:

QNX Momentics Tool Suite

A comprehensive set of integrated development tools, lots of in-depth documentation, as well as powerful diagnostics and optimization tools for your system once it's up and running on your target.

QNX Neutrino RTOS

Trusted and proven in countless embedded systems, QNX Neutrino has a growing reputation as the world's most reliable RTOS.

We now invite you to explore the advanced tools that the QNX Momentics Tool Suite adds to QNX Neutrino.

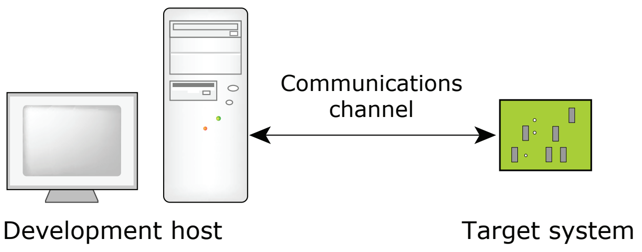
Choice, tools, source, and help

As a complete package designed for embedded systems developers, the QNX Software Development Platform gives you everything you need at every stage of your product-development cycle:

Choice	Choice of host (Windows, Linux, QNX Neutrino), choice of target (ARM, MIPS, PowerPC, SH-4, XScale, x86), and choice of development language (C, C++, Embedded C++, Java).
Tools	Code development, editors, source control, compilers, libraries, profilers, analyzers, optimizers, etc.
Help	Documentation, forums, technical support programs, etc.

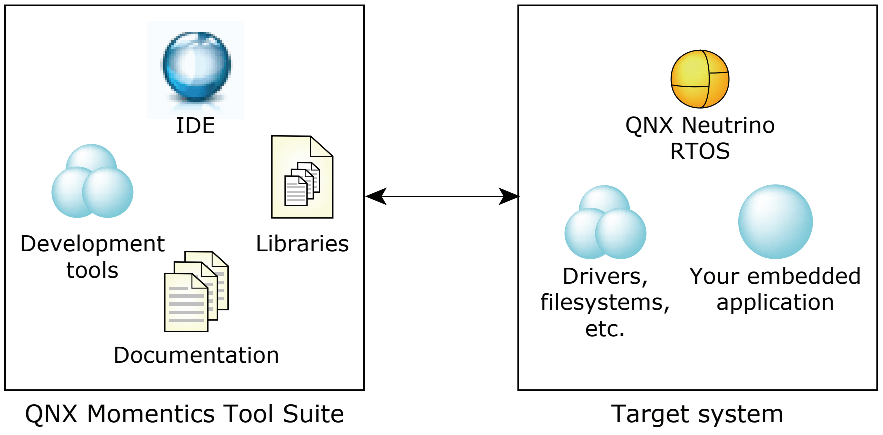
Full source code for numerous startup programs, IPLs, device drivers, etc. is available from our website.

You can install QNX SDP on a QNX Neutrino system for *self-hosted development*, or you can install it on a Windows or Linux *development host* and install QNX Neutrino on a *target system*:



Development host and target system.

The development host runs the QNX Momentics Tool Suite; the target system runs the QNX Neutrino RTOS itself plus all the programs you develop:



QNX Momentics is the development environment on your host for the QNX Neutrino RTOS running on your target.

QNX Momentics at a glance

If the QNX Neutrino RTOS is the “engine” that will empower the embedded system you’re developing, then QNX Momentics is the “factory” where you modify your engine as well as build, test, and finish your vehicles.

Here are the main parts of the QNX Momentics Tool Suite:

Integrated Development Environment

This is your toolbox on Linux and Windows. The IDE’s task-oriented interface helps you quickly set up your project, choose your programming language, choose a target processor, compile your code, connect to your target, transfer your application to your target, run it, debug it, profile it, and fine-tune it.

Command-line tools

If you aren't using the IDE, you can use command-line tools to develop applications. For example, you can use **qcc** to compile and link, and **mkifs** to create an OS image.

Libraries — ANSI C, POSIX, Dinkum C++ (full and embedded), GNU C++ (x86 only), graphics, widgets, compression, etc.

Documentation — How-to guides, references, context-sensitive help, and technotes. See the chapter How to Get Help to help you find your way through the documentation.

Additional components

Once you've installed the QNX Software Development Platform, you can download these components from our website (**http://www.qnx.com/**) after logging into your myQNX account:

Board Support Packages

Software and step-by-step instructions to help you get Neutrino and your applications running on specific boards.

Driver Development Kits

Full source and detailed documentation to help you write your own drivers for various devices: audio, graphics, input (mice, keyboards, etc.), and USB.

Note that the QNX Software Development Platform includes the DDK documentation.

QNX Aviage middleware

The QNX Aviage brand encompasses a portfolio of middleware products that help you create consumer-grade audio, video, and graphical products in record time. QNX Aviage software products accelerate your innovation by offering the following:

QNX Aviage Acoustic Processing

A sophisticated acoustic hands-free and speech-enhancement solution designed specifically for challenging automotive environments. Unlike traditional voice-quality enhancement solutions, it eliminates the need for dedicated hardware, lowers production costs, and increases design flexibility.

QNX Aviage HMI Suite

Allows embedded developers to implement easy-to-use and compelling user interfaces in Adobe Flash, thereby eliminating the lengthy process of coding HMIs with graphical APIs or cumbersome tool kits.

QNX Aviage Multimedia Suite

Configurable, fully featured media jukebox with supporting multimedia software for building next-generation digital infotainment platforms. Includes powerful APIs for quickly building multi-modal HMIs. Options include iPod support, Microsoft PlaysForSure support (with DRM), as well as an array of software codecs.

For more information about QNX Aviage middleware, contact your sales representative.

Getting started *before* you have your target

Assuming you plan to develop a Neutrino-based embedded system of some kind (e.g. a vehicle telematics system, a router, a medical imaging device), you can start developing your application, even before you have your target hardware.

And if you haven't yet decided on the CPU family for your target (e.g. PowerPC, XScale), you can still begin developing your application. As a rule, with QNX Momentics you write your application code once, then compile it however many times you need for whatever supported targets you're using. As you'll see if you plan to use the IDE, you can build the very same project for any of our supported processors with just a couple of mouse clicks.

If you're developing on a self-hosted Neutrino system, you can run your application directly on your development machine.

Consider the following scenarios:

x86 (PC) target

If your final target hardware will be a PC in some shape or form (e.g. a PC/104 SBC module), you can simply connect your host to any spare PC and use that as your temporary target. You should expect to make very few changes (if any) in your application when the time comes to move to your final target.

And if you don't happen to have a spare PC, you could use a *virtual machine* — products such as VMware Workstation and VMware Player (`http://www.vmware.com`), and Microsoft VirtualPC 2007 can emulate a separate, complete hardware environment right on your desktop.

Once you've set up a virtual machine (VM) on your host, you can then use it as your target — you can run, test, and debug your Neutrino applications on the VM as if it were an actual machine connected to your host.

Virtual machines don't necessarily support hard realtime.

x86 non-BIOS target

In this case, the target computer (e.g. an LX800) isn't a PC. It doesn't have a BIOS, so it won't boot and load Neutrino in the same way as a PC. In order to run Neutrino on such a target, you need an appropriate BSP for your particular hardware, including an IPL that could set up the processor and load Neutrino properly.

We're continually adding new BSPs to our extensive selection; for a list, see the Products area of our website, **`http://www.qnx.com`**. If you don't see a BSP for your target hardware, contact us.

You can develop your application without target hardware, because much of your code will be hardware-independent. For device drivers and other hardware-specific areas of your system, you may be able to use a virtual platform tool that accurately emulates your specific embedded system hardware.

For instance, Virtio (**`http://www.virtio.com`**) makes virtual platforms that emulate various processors and peripherals, right down to the CPU's instruction set. You can download a free time-limited evaluation copy from the vendor's website.

Any supported CPU family

Whatever your target hardware, you can still go a long way down your development path using QNX Momentics right now. Develop all your application-level components first, leaving device drivers and other hardware-specific details until you have your target hardware.

As mentioned earlier, you might want to consider Virtio if you need a functionally accurate virtual prototype of your embedded platform before you have the actual hardware itself.

Platform-related issues

Whatever the specific CPU family, your application code will, in most cases, run unchanged when you go to your final target. But you may need to be aware of a few platform-related issues (e.g. endian differences, CPU speeds, I/O addressing, alignment of data structures, memory limitations, synchronization on multicore vs uniprocessor systems, etc.).

Fortunately, QNX Momentics includes several header files and convenience functions that will help you anticipate and easily handle such problems.

For more information on these and other platform-related issues, see Freedom from Hardware and Platform Dependencies in the Neutrino *Programmer's Guide*.

A word about licensing

Here are some general questions about licensing. For more information, visit the **Licensing** area of **`http://www.qnx.com`**, where you'll find details on our licensing model, types of licenses, etc., or email **`licensing@qnx.com`**.

Where can I find my EULA?

You'll find your Momentics End User License Agreement and the full license guides on your DVD and also in the Licensing area of our website, **`http://www.qnx.com`**

Once you've installed the QNX Software Development Platform, you can find the license agreements in *base_dir*/**`install/qnxsdp/`***version*, where *base_dir* is where you installed SDP (see the output from the **`qconfig`** command).

I've installed the QNX Software Development Platform on my workstation. Can I also make a copy for my laptop?

Certainly. Our licensing policy is per *seat* (i.e. per person), not per machine. You may use QNX SDP on your primary workstation as well as on a laptop or on a PC at your home, provided that you use only one system at any given time.

Can I get a Neutrino runtime system?

We don't provide a Neutrino runtime system, but you can build one (as permitted by your license agreement) that includes your own applications. For more information, see *How to create a Runtime Kit from the QNX Software Development Platform* in the QNX Neutrino Technotes.

As an OEM, how do I get a runtime license?

To obtain a runtime license (which governs how you may distribute certain Neutrino runtime files as part of your product), please contact your QNX sales representative.

Chapter 2

Getting Started

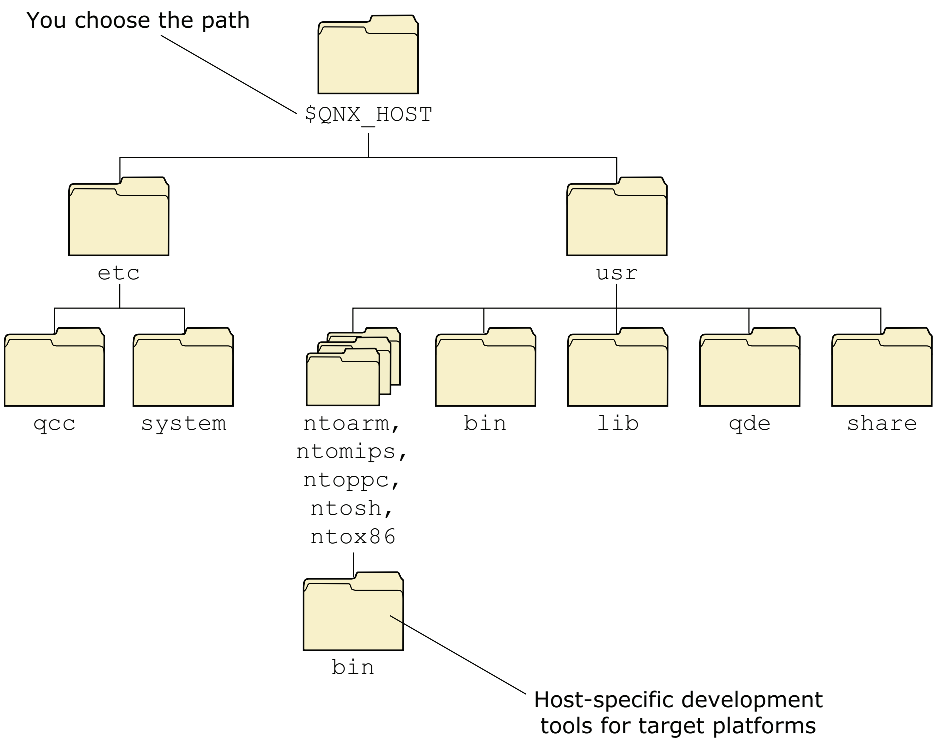
In this chapter. . .

How QNX Momentics is organized

The QNX Momentics Tool Suite is organized around these two main areas:

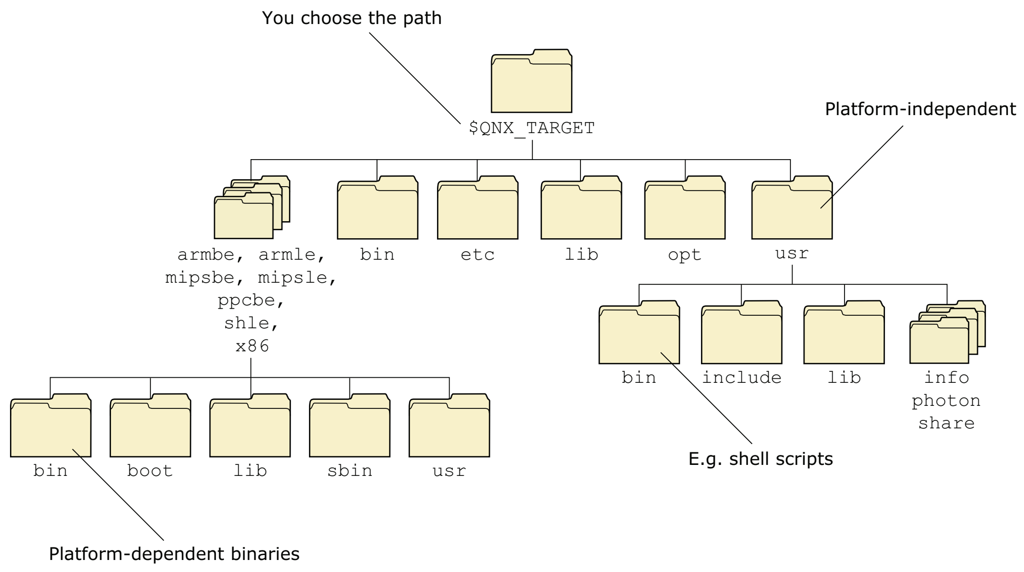
Host-related	All your libraries, executables, etc., designed to run on your host system (e.g. Windows).
Target-related	All CPU-specific components, as well as certain common things that you can use on any target system.

The **QNX_HOST** environment variable identifies the directory that holds the host-related components:


You choose the path
$QNX_HOST
etc
usr
qcc
system
ntoarm, ntomips, ntoppc, ntosh, ntox86
bin
lib
qde
share
bin
Host-specific development tools for target platforms

The host-related directory structure.

The **QNX_TARGET** environment variable identifies the directory that holds the target-related components:



The target-related directory structure.

Neutrino also uses these environment variables to locate files on the host machine:

QNX_CONFIGURATION
The location of the configuration files and licenses for QNX Momentics.

MAKEFLAGS The location of included **`*.mk`** files.

Here's where some of the key components are installed:

Component	Location
Buildfiles	`$QNX_TARGET/`*platform*`/boot/build/`*boardname*`.build`
Command-line utilities	For the host: `$QNX_HOST/usr/bin` and `$QNX_HOST/`*platform*`/bin` For the target: `$QNX_TARGET/bin` and `$QNX_TARGET/`*platform*`/bin` and `$QNX_TARGET/`*platform*`/sbin`
Device drivers (binaries)	`$QNX_TARGET/`*platform*`/sbin`
Device drivers (DLLs)	`$QNX_TARGET/`*platform*`/lib/dll`
Filesystems	`$QNX_TARGET/`*platform*`/sbin`
GUI-related	`$QNX_TARGET/usr/photon`
Shared libraries	`$QNX_TARGET/`*platform*`/lib`
System header files	`$QNX_TARGET/usr/include`
Documentation	Eclipse plugin directory, also in `$QNX_TARGET/usr/help/product` on self-hosted Neutrino systems

If you install any BSPs or DDKs, they're installed under **`${QNX_TARGET}`**, or in a directory of your choosing. The IDE creates a *workspace*, a directory that holds your development projects. By default, this workspace is in your home directory on Linux, and in **`C:\QNX640`** on Windows.

For information about the directory structure on a Neutrino runtime system, see "Where everything is stored" in the Working with Files chapter of the Neutrino *User's Guide*.

What's on my desktop?

How you access the components of QNX Momentics depends on your host machine:

Windows

You can start the IDE by clicking its icon on the desktop:

or by choosing **QNX Momentics IDE** from the **Start→All Programs→QNX Software Development Platform** menu. This menu also lets you add or activate licenses, configure your machine to build for a specific version of QNX Neutrino, run Phindows, and start the IDE.

Linux

The **Start→Programming** menu lets you add or activate licenses, and start the IDE. You can also start the IDE by running the **`qde`** command.

Neutrino	The **Launch→Configure** menu lets you add and activate licenses. The Help item in the Launch menu and on the shelf starts the Helpviewer, where you'll find the documentation.

Upgrading your software

Here's the general procedure for updating a version of the QNX Software Development Platform or other components you've purchased:

1. Go to the QNX Software Systems website (`http://www.qnx.com`) and log into your myQNX account. If you don't already have a myQNX account, please register now.
2. Follow the instructions for registering your product. You'll need the Product Registration serial number and password, which you'll find in the box that contains the installation disks.

   For more information about setting up your myQNX account, see *Accessing Online Technical Support*. There's a printed copy in the QNX Software Development Platform box, and a PDF version on the DVD and on our website.
3. Go to the Download area.
4. Select the product (e.g. "QNX Software Development Platform") or search by keywords.

In the next step, you'll download a file. Don't download it into a directory whose path contains spaces. For example, don't download the file into `C:\Documents and Settings\my_userid\Desktop`.

5. Download the appropriate file and follow the instructions.

If you installed an evaluation copy of the QNX Software Development Platform, you can upgrade to a permanent copy without reinstalling. For more information, see the *Installation Guide*.

Managing source code

You'll probably want to use some sort of version-control system to manage and track changes to the software that you develop.

For information about:	**See:**
CVS (Concurrent Versions System)	Using CVS in the Neutrino *User's Guide*, and Managing Source Code in the IDE *User's Guide*.

continued...

For information about:	**See:**
Subversion (svn)	Collins-Sussman, Ben, Fitzpatrick, Brian W., Pilato, C. Michael. 2004. *Version Control with Subversion.* Sebastopol, CA: O'Reilly & Associates. ISBN: 9780596004484

QNX Momentics includes clients for both CVS and Subversion.

Running QNX Neutrino self-hosted

You can develop software on a self-hosted QNX Neutrino system. For more information on working with Neutrino, see the Neutrino *User's Guide*; for information on developing software on Neutrino, see the Neutrino *Programmer's Guide* and the IDE *User's Guide*.

Mixing a self-hosted machine with other hosts

If you have a Neutrino host, you can communicate with other hosts in various ways:

- You can access resources — such as files, directories, and processes — on other Neutrino machines as if the resources were on your own computer; see Using Qnet for Transparent Distributed Processing in the Neutrino *User's Guide*.
- You can use TCP/IP; see TCP/IP Networking in the Neutrino *User's Guide*.
- You can mount DOS and Linux filesystems right on your Neutrino box, or use CIFS or NFS to mount filesystems across a network; see Working with Filesystems in the Neutrino *User's Guide*.

Can different versions of QNX Momentics coexist?

The QNX Momentics Tool Suite lets you install and work with multiple versions of Neutrino (from 6.2.1 and later). Whether you're using the command line or the IDE, you can choose which version of the OS to build programs for.

Coexistence with 6.2.1 is supported only on Windows hosts.

When you install QNX Momentics, you get a set of configuration files that indicate where you've installed the software. The **QNX_CONFIGURATION** environment variable stores the location of the configuration files for the installed versions of Neutrino; on a self-hosted Neutrino machine, the default is **`/etc/qnx`**.

`QWinCfg` for Windows hosts

On Windows hosts, you'll find a configuration program called **`QWinCfg`** for switching between versions of QNX Momentics. You launch **`QWinCfg`** via the start menu (e.g. **All Programs→QNX Software Development Platform 6.5.0→Configuration**).

For details on using **`QWinCfg`**, see its entry in the *Utilities Reference*.

`qconfig` utility for non-Windows hosts

If you're using the command-line tools, use the **`qconfig`** utility to configure your machine to use a specific version of Neutrino:

- If you run it without any options, **`qconfig`** lists the versions that are installed on your machine.
- If you specify the **`-e`** option, you can set up the environment for building software for a specific version of the OS. For example, if you're using the Korn shell (**`ksh`**), you can configure your machine like this:

```
eval ‘qconfig -n "QNX Software Development Platform 6.5.0" -e‘
```

In the above command, you must use the "back tick" character (**`‘`**), *not* the single quote character (**`'`**). The string that you pass to the **`-n`** option is the Installation Name field as printed by **`qconfig`**.

This command affects only the shell in which you ran **`qconfig`**. Other windows, for example, will be unaffected. To change environments in all your windows, you can run the command in your shell-initialization script or in your **`.profile`**. You can also define separate users who use different coexisting versions.

For more information, see the Compiling and Debugging chapter of the Neutrino *Programmer's Guide*.

Coexistence and the IDE

When you start the IDE, it uses your current **`qconfig`** choice as the default version of the OS; if you haven't chosen a version, the IDE chooses an entry from the directory identified by **QNX_CONFIGURATION**. If you want to override the IDE's choice, you can choose the appropriate build target. For more information, see the IDE Concepts chapter of the IDE *User's Guide*.

Running QNX Neutrino on a target machine

Neutrino is well suited to embedded systems. For information about creating OS images, downloading them to your target hardware, and running your software, see:

- the BSP documentation for your particular target (in the IDE's help system, or in the Photon Helpviewer on Neutrino)
- *Building Embedded Systems*
- IDE *User's Guide*

You don't always need to have the hardware to run your software; for more information, see "Getting started *before* you have your target" in the What is the QNX Software Development Platform? chapter in this guide.

Chapter 3

How to Get Help

In this chapter. . .

The first place to look for help is in our documentation, but if you still have problems, there are several other avenues of help.

Overview of the documentation

In the QNX Software Development Platform, the online documents are in HTML, which you can access in the IDE's help system. On self-hosted QNX Neutrino systems, you can also look at the documentation in the Photon helpviewer. You can download PDF versions of the documentation from our website, **`http://www.qnx.com`**.

The complete QNX SDP documentation set contains the following books, arranged here under each main component:

QNX Software Development Platform

Aside from the *Welcome to the QNX Software Development Platform* guide, this bookset includes:

10 Steps to Developing a QNX Program: Quickstart Guide	A tutorial that helps you install QNX Momentics on a host machine, install the QNX Neutrino RTOS on a target machine, set up communications between the two systems, and then use the IDE to develop a program on the host machine and run it on the target.
Installation Guide	Instructions for installing and uninstalling QNX SDP.
Release Notes	Important details about QNX SDP. For the most up-to-date version of these notes, go to our website, **`http://www.qnx.com`**.

QNX Momentics Tool Suite

Utilities Reference	Describes the Neutrino configuration files, utilities, and manager processes you'll use during development and at runtime.
IDE *User's Guide*	Describes the QNX Momentics Integrated Development Environment, how to set up and start using the tools to build Neutrino-based target systems, etc.
Phindows Connectivity	Tells you how to access Photon from a Windows machine.

QNX Neutrino Realtime Operating System

System Architecture	Describes the concepts and architecture of the QNX Neutrino microkernel, resource managers, processes, threads, message-passing services, and more.
QNX Neutrino *User's Guide*	Explains how to interact with a running Neutrino system. Covers both Photon and text-mode interfaces, as well as various system-administration topics.
Getting Started with QNX Neutrino: A Guide for Realtime Programmers	This book, by Rob Krten, will help you design and develop robust realtime systems — from tiny embedded control applications to large network-distributed systems — using the QNX Neutrino RTOS.
QNX Neutrino *Programmer's Guide*	Tells you how to get started writing programs, including interrupt handlers, etc.
Building Embedded Systems	Tells you how to get the OS running on your target embedded system, write an IPL, customize a startup program, etc.
Utilities Reference	Describes the Neutrino configuration files, utilities, and manager processes you'll use during development and at runtime.
QNX Neutrino *Library Reference*	Describes the C library data types and functions, including POSIX threads, kernel calls, resource manager functions, etc.
Writing a Resource Manager	Describes how to write a program that registers a name in the filesystem name space, which other processes then use to communicate with the resource manager. A resource manager is frequently (but not always) a device driver.
Audio Developer's Guide	Describes how to write audio applications.
Addon Interfaces Library Reference	Describes the Addon Interfaces Library and how to use it to add extendability to applications that use standard interfaces.
Adaptive Partitioning *User's Guide*	Describes how to divide system resources (e.g. processor time) in a flexible way between competing processes.

Core Networking *User's Guide*
: Describes the QNX Neutrino Core Networking stack and its manager, **`io-pkt`**.

High Availability Framework *Developer's Guide*
: Describes how to use the High Availability Manager (HAM) to detect, isolate, and recover from software faults.

Instant Device Activation *User's Guide*
: Describes how to start devices quickly when the system boots.

Multicore Processing *User's Guide*
: Describes how to get the most performance possible out of a multicore system.

Persistent Publish/Subscribe Developer's Guide
: Information about the Persistent Publish/Subscribe (PPS) system, a resource manager that makes it easy to disseminate information to interested processes.

System Analysis Toolkit *User's Guide*
: Describes how to use the SAT with our instrumented microkernel. You can log every communication and state change within the microkernel, including interrupts, all parameters/return values from kernel calls, and scheduling decisions, resulting in a deeper and more detailed analysis of system elements. You can even perform kernel-level diagnostics remotely.

BSP guides
: Describe how to get Neutrino running on your target board. We support boards in these processor families: ARM/XScale, MIPS, PowerPC, SH-4, and x86. You can download BSPs from our Foundry27 website,

  `http://community.qnx.com`

Driver Development Kit guides
: Describe how to write drivers for QNX Neutrino. You'll find a separate DDK guide for audio, character, graphics, input, and Universal Serial Bus (USB) devices. You can download the DDKs from our website, but the QNX Software Development Platform includes the documentation.

Technical Notes
: Deals with a series of topics (e.g. IP tunneling) that aren't covered in the basic documentation set.

Photon microGUI

Photon *Programmer's Guide*

Gives you a hands-on tour of the Photon Application Builder (PhAB). You'll learn how to quickly assemble a GUI from predefined widgets, link the GUI to an application, and generate C source to bring the GUI to life.

Photon *Library Reference*

Provides concise descriptions of Photon's and PhAB's global data structures and functions.

Widget Reference

Contains guidelines for programming widgets, along with concise descriptions of all global data structures, resources, and convenience functions associated with widgets. It also gives you practical examples of how to use Photon widgets and widget functions.

Building Custom Widgets

Explains how to create a custom widget and how to bind it into PhAB. If you need a widget whose features extend the standard capabilities of the Photon widget library, this guide is for you.

Photon Multilingual Input

Tells you how to input Chinese, Japanese, and Korean characters in Photon.

Advanced Graphics

Advanced Graphics *Developer's Guide*

Describes how to create graphical applications that use the QNX Graphics Framework API and the OpenGL ES APIs.

Composition Manager *Developer's Guide*

Describes how to use the QNX Composition Manager, a hardware-independent layer of abstraction that encompasses all aspects of window management, such as window creation, realization, and destruction.

Web Browser Engine *Developer's Guide*

Describes how to build a web browser using the Web Browser Engine, a high-performance embeddable web browser that's based on the WebKit open-source engine.

Dinkum C and C++

Dinkum C++ Library

A conforming implementation of the Standard C++ library.

Dinkum C99 Library

A conforming implementation of the Standard C library, as revised in 1999.

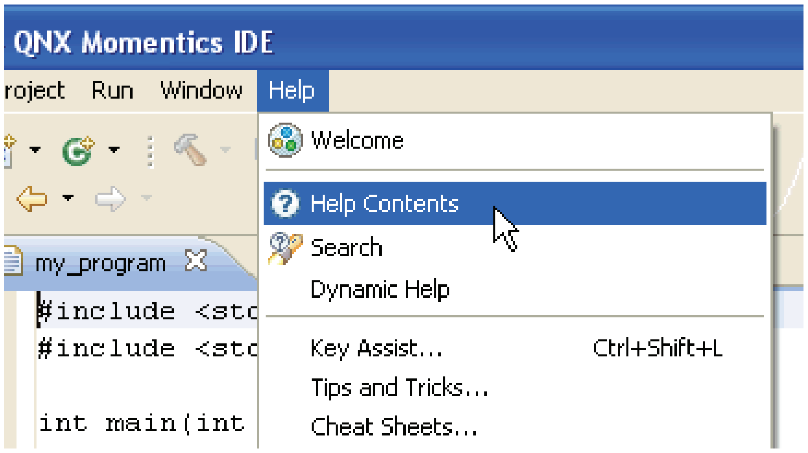
Dinkum EC++ Library

A conforming implementation of the Embedded C++ library as specified by the Embedded C++ Technical Committee.

Viewing the documentation

Within the IDE

Click **Help→Help Contents**. There you'll find several booksets listed, including *A Roadmap to the QNX Software Development Platform*. The other documents listed, such as the *Workbench User Guide* and *JDT Plug-in Developer Guide*, pertain to the Eclipse platform and its various plugins.



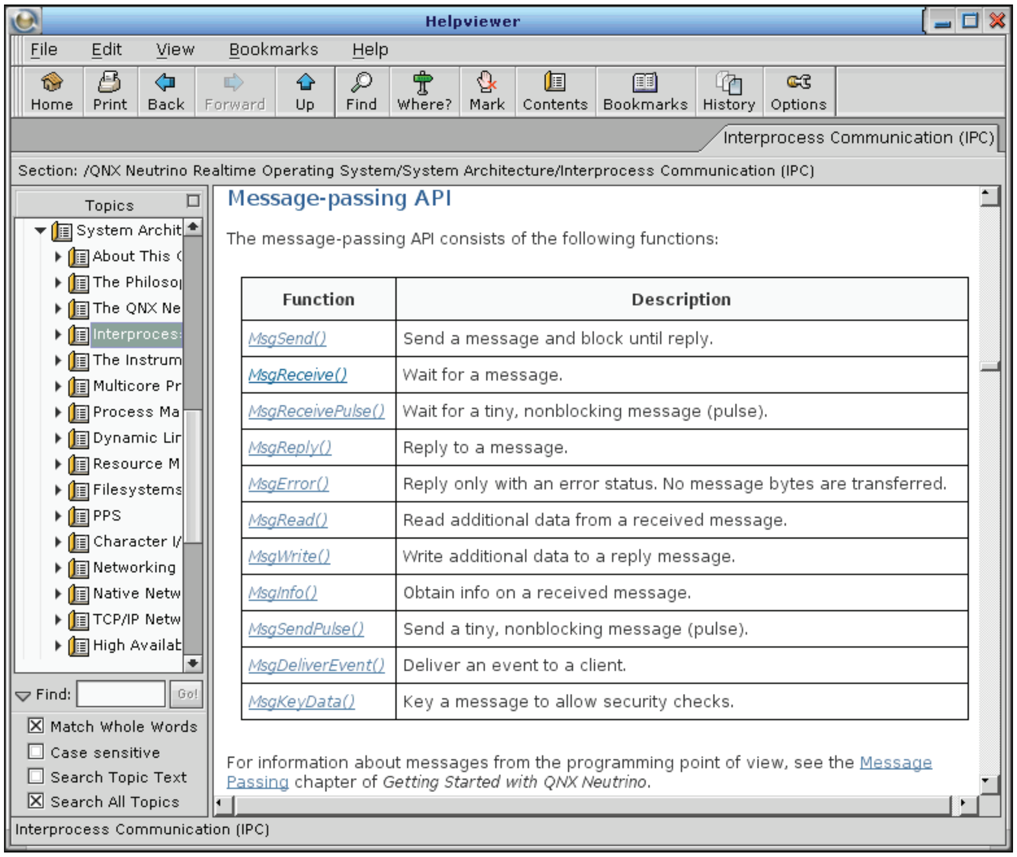
```
#include <std
#include <std

int main(int
```

Getting help in the IDE.

Note that the IDE's internal help system has a builtin search facility. For details, see "Using the QNX Help system" in the Getting Started chapter of the IDE *User's Guide*.

The Photon Helpviewer

If you're using Neutrino self-hosted, you can access the documentation via our native Helpviewer.


Helpviewer
File Edit View Bookmarks Help
Home Print Back Forward Up Find Where? Mark Contents Bookmarks History Options
Interprocess Communication (IPC)
Section: /QNX Neutrino Realtime Operating System/System Architecture/Interprocess Communication (IPC)
Topics
System Archit
About This
The Philoso
The QNX Ne
Interproces
The Instrum
Multicore Pr
Process Ma
Dynamic Lir
Resource M
Filesystems
PPS
Character I/
Networking
Native Netw
TCP/IP Netw
High Availab
Find:
Go!
Match Whole Words
Case sensitive
Search Topic Text
Search All Topics
Interprocess Communication (IPC)

Message-passing API

The message-passing API consists of the following functions:

Function	Description
MsgSend()	Send a message and block until reply.
MsgReceive()	Wait for a message.
MsgReceivePulse()	Wait for a tiny, nonblocking message (pulse).
MsgReply()	Reply to a message.
MsgError()	Reply only with an error status. No message bytes are transferred.
MsgRead()	Read additional data from a received message.
MsgWrite()	Write additional data to a reply message.
MsgInfo()	Obtain info on a received message.
MsgSendPulse()	Send a tiny, nonblocking message (pulse).
MsgDeliverEvent()	Deliver an event to a client.
MsgKeyData()	Key a message to allow security checks.

For information about messages from the programming point of view, see the Message Passing chapter of *Getting Started with QNX Neutrino*.

Photon Helpviewer.

To open the Helpviewer, click the **Help** button in the Applications group on the shelf or select **Help** from the right-click menu on the desktop.

For more information, see “Getting help with the Helpviewer” in the Using the Photon microGUI chapter of the Neutrino *User’s Guide*, as well as the entry for **`helpviewer`** in the *Utilities Reference*.

Keyword indexes

Nearly every book in the QNX documentation set has its own keyword index. At the top and bottom of the online documents, you’ll find a link to the keyword index file (**`keywords-all.html`**).

What should I read first?

Many people simply don’t read manuals cover to cover. They often browse or search the documentation for a specific topic, usually for information on how to do a certain task. But if you want to approach your tasks with enough knowledge to work effectively, it’s a good idea to start with the *System Architecture* guide; it will help you understand Neutrino's unique features, particularly its message-based interprocess communication (IPC) and microkernel architecture.

Once you know how Neutrino works, you'll then want to know how to work with it; our *10 Steps to Developing a QNX Program: Quickstart Guide* is a short tutorial that will get you started in a matter of minutes. Which document you'll need next depends on when you need it.

During development (on your host)

Most of the documents in the bookset are geared towards developing your Neutrino-based applications. Of these development books, some are how-to guides, and some are reference works.

Some books are useful during development as well as at runtime. For instance, the *Utilities Reference* is a comprehensive document that includes descriptions of both development utilities (e.g. **`make`**), which you wouldn't normally use on your target, as well as runtime programs (e.g. **`devc-sersci`**), which you would run only on your target.

Here are the main how-to guides and their corresponding reference books:

How-to guide:	**Companion reference:**
IDE *User's Guide*	*Utilities Reference* (for underlying command-line utilities, such as **`make`**); QNX Neutrino *Library Reference* for API
QNX Neutrino *User's Guide*	*Utilities Reference*
Getting Started with QNX Neutrino: A Guide for Realtime Programmers, QNX Neutrino *Programmer's Guide*	QNX Neutrino *Library Reference*
Photon *Programmer's Guide*	Photon *Library Reference*; Photon *Widget Reference*
Building Embedded Systems	*Utilities Reference*; QNX Neutrino *Library Reference*
Writing a Resource Manager	QNX Neutrino *Library Reference*

At runtime (on your target)

The most runtime-oriented document in your bookset is the QNX Neutrino *User's Guide*, which describes how to use and interact with a running Neutrino system. The book covers both Photon and text-mode interfaces, as well as various system-administration topics.

Other runtime-oriented documents include the *Utilities Reference* and the IDE *User's Guide* (for information on diagnostic tools you'd use on a running system).

Where key features are documented

The following list may help you learn which document to look in when you want information on certain key features or components of the OS and tools.

See the *System Architecture* guide for information on almost every topic in this list.

Adaptive partitioning	Adaptive Partitioning *User's Guide*
Asymmetric multiprocessing (AMP)	Multicore Processing in the *System Architecture* guide
Audio support	*Audio Developer's Guide* (for writing audio applications); *Audio DDK* (for writing audio device drivers); `io-audio` manager and the `deva-*` drivers in the *Utilities Reference*.
Bound multiprocessing (BMP)	Multicore Processing in the *System Architecture* guide; the Multicore Processing *User's Guide*; `procnto*` in the *Utilities Reference*.
BSPs	See our Foundry27 website, `http://community.qnx.com`
Buildfiles	Making an OS Image in *Building Embedded Systems*; `mkifs` in the *Utilities Reference*.
CD-ROM	Working with Filesystems in the QNX Neutrino *User's Guide*; `cam-cdrom.so` in the *Utilities Reference*.
Channels	*Channel**, *Msg**, and *Connect** functions in the QNX Neutrino *Library Reference*; Message Passing in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; *Writing a Resource Manager*.
Compiling	*10 Steps to Developing a QNX Program: Quickstart Guide*; Developing C/C++ Programs in the IDE *User's Guide*; Compiling and Debugging in the QNX Neutrino *Programmer's Guide*; `make` and `qcc` in the *Utilities Reference*.
Compression	Making an OS Image in *Building Embedded Systems*; `deflate` in the *Utilities Reference*.
CPU time, sharing among competing processes	Adaptive Partitioning *User's Guide*
Data server	Setting Up an Embedded Web Server in the QNX Neutrino *User's Guide*; `ds` in the *Utilities Reference*.

DDKs	Driver Development Kits (DDKs). The QNX Software Development Platform includes the documentation, but if you want the DDKs, you must download them from our website.
Debugger	*10 Steps to Developing a QNX Program: Quickstart Guide*; Debugging Programs in the IDE *User's Guide*; Compiling and Debugging in the QNX Neutrino *Programmer's Guide*; **`gdb`** and other debugging utilities in the *Utilities Reference*.
Device management	*devctl()* in the QNX Neutrino *Library Reference*; Resource Managers in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; *Writing a Resource Manager*; **`dev*`** drivers in the *Utilities Reference*; Instant Device Activation *User's Guide*.
Editors	*10 Steps to Developing a QNX Program: Quickstart Guide*; IDE Concepts in the IDE *User's Guide*; Using Editors in the QNX Neutrino *User's Guide*; **`elvis`**, **`ped`**, **`qed`**, **`sed`**, and **`vi`** in the *Utilities Reference*.
Embedded systems	*Building Embedded Systems*.
Endian issues	Freedom from Hardware and Platform Dependencies in the QNX Neutrino *Programmer's Guide*.
Environment, configuring	Controlling How Neutrino Starts and Configuring Your Environment chapters in the QNX Neutrino *User's Guide*.
Environment variables	Commonly Used Environment Variables appendix in the *Utilities Reference*; *environ()* in the QNX Neutrino *Library Reference*; Using the Photon microGUI and Configuring Your Environment chapters in the QNX Neutrino *User's Guide*.
Event handling (OS)	*MsgDeliverEvent()* in the QNX Neutrino *Library Reference*; Message Passing in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; Writing an Interrupt Handler in the QNX Neutrino *Programmer's Guide*.
Event handling (Photon)	Events in the Photon *Programmer's Guide*; **`phrelay`** in the *Utilities Reference*.
Faults, recovering from	High Availability Framework *Developer's Guide*.
File handling	*creat()* and related functions via its "See also" section in the QNX Neutrino *Library Reference*; **`basename`**, **`cat`**, etc. in the *Utilities Reference*.

Filesystems	Working with Filesystems in the QNX Neutrino *User's Guide*.
Fonts	Fonts chapter and Photon in Embedded Systems appendix in the Photon *Programmer's Guide*.
Graphics	*Graphics DDK*; **`devg-*`** drivers in the *Utilities Reference*; Photon *Programmer's Guide*; Advanced Graphics *Developer's Guide*; Composition Manager *Developer's Guide*
Hardware	Connecting Hardware in the QNX Neutrino *User's Guide*.
HID (human input devices)	**`hidview`** and **`devh-*`** drivers in the *Utilities Reference*.
Helpviewer (Photon)	Using the Photon microGUI in the QNX Neutrino *User's Guide*; **`helpviewer`** in the *Utilities Reference*.
I/O management	*iofunc**, *Interrupt**, and *open()* and related functions via *open()*'s "See also" section in the QNX Neutrino *Library Reference*; Resource Managers in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; *Writing a Resource Manager*.
IPC (interprocess communication)	*pthread_mutex**, *SyncMutex**, *sem_**, *SyncSem**, *pthread_cond**, and *SyncCondvar** in the QNX Neutrino *Library Reference*. See also "Message Passing," below.
IPL (Initial Program Loader)	Writing an IPL Program in *Building Embedded Systems*.
Images (OS)	Making an OS Image in *Building Embedded Systems*; **`mkifs`** in the *Utilities Reference*.
Images (graphical)	Raw Drawing and Animation in the Photon *Programmer's Guide*.
Input	*Input DDK*; **`devh-*`** and **`devi-*`** drivers in the *Utilities Reference*.
Instrumented kernel	System Analysis Toolkit *User's Guide*.
Interrupt handling	*InterruptAttach()* and related functions in its "See also" section in the QNX Neutrino *Library Reference*; Interrupts in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; Writing an Interrupt Handler in the QNX Neutrino *Programmer's Guide*.
Kernel calls	*Channel**, *Clock**, *Connect**, *Debug**, *Interrupt**, *Msg**, *Sched**, *Signal**, *Sync**, *Thread**, *Timer**, and *TraceEvent()* in the QNX Neutrino *Library Reference*.

Keyboard support	Using the Command Line in the QNX Neutrino *User's Guide*; **`mkkbd`** and related utilities in its "See also" section in the *Utilities Reference*.
Libraries	Compiling and Debugging in the QNX Neutrino *Programmer's Guide*; Building OS and Flash Images in the IDE *User's Guide*.
Linker	Compiling and Debugging in the QNX Neutrino *Programmer's Guide*; **`qcc`** and related utilities in its "See also" section in the *Utilities Reference*.
`Makefile` structure	Developing C/C++ Programs in the IDE *User's Guide*; Conventions for Recursive Makefiles and Directories in the QNX Neutrino *Programmer's Guide*.
Memory management	*mem**, *mmap**, *posix_mem**, and *malloc()* in the QNX Neutrino *Library Reference*; Heap Analysis: Making Memory Errors a Thing of the Past in the QNX Neutrino *Programmer's Guide*; Finding Memory Errors in the IDE *User's Guide*.
Message passing	*Msg**, *Connect**, and *Channel** in QNX Neutrino *Library Reference*. See also "IPC," above.
Message queues	*mq_** functions in the QNX Neutrino *Library Reference*; **`mqueue`** and **`mq`** in the *Utilities Reference*.
Multicore systems	Multicore Processing in the *System Architecture* guide; the Multicore Processing *User's Guide*; **`procnto*`** in the *Utilities Reference*.
Native networking (Qnet)	See "Transparent distributed processing," below.
Network drivers	**`io-pkt*`**, **`devn-*`**, **`devnp-*`** drivers in the *Utilities Reference*; QNX Neutrino Core Networking *User's Guide*
Permissions (on files, directories)	Managing User Accounts and Working with Files chapters in the QNX Neutrino *User's Guide*; *chmod()*, *stat()*, and *umask()* in the QNX Neutrino *Library Reference*; **`chmod`** in the *Utilities Reference*.
Process manager	**`procnto*`** in *Utilities Reference*; *procmgr_** in the QNX Neutrino *Library Reference*.
Processes	Processes and Threads in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; Processes in the QNX Neutrino *Programmer's Guide*; *exec**, *fork**, and *spawn** in the QNX Neutrino *Library Reference*.

Profiler (application)	Profiling an Application in the IDE *User's Guide*.
Profiler (system)	Analyzing Your System with Kernel Tracing in the IDE *User's Guide*.
Pulses	*MsgSendPulse()*, *MsgReceivePulse()*, and *pulse_** in the QNX Neutrino *Library Reference*; Message Passing in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; "Handling private messages and pulses" in the Handling Other Messages chapter of *Writing a Resource Manager*.
QoS (Quality of Service)	"Quality of Service and multiple paths" in the Native Networking (Qnet) chapter in *System Architecture*; *netmgr_ndtostr()* in the QNX Neutrino *Library Reference*.
RAM disk	**`io-blk.so`** (using the **`ramdisk=`***size* option), **`devb-ram`**, and **`devf-ram`** in the *Utilities Reference*.
Realtime scheduling	*Sched** functions and **`sched_param`** structure in the QNX Neutrino *Library Reference*.
Resource management	Resource Managers in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; *Writing a Resource Manager*; *resmgr_** and *iofunc_** in the QNX Neutrino *Library Reference*.
Resources, sharing among competing processes	Adaptive Partitioning *User's Guide*
Symmetric multiprocessing (SMP)	Multicore Processing in the *System Architecture* guide; the Multicore Processing *User's Guide*; **`procnto*`** in the *Utilities Reference*.
Self-hosted development	Compiling and Debugging in the QNX Neutrino *Programmer's Guide*.
Signal handling	*sigaction()* and related functions via its "See also" section in the QNX Neutrino *Library Reference*.
Startup programs	Customizing Image Startup Programs in *Building Embedded Systems*; **`startup-*`** in the *Utilities Reference*.
Synchronization	*Sync** functions in the QNX Neutrino *Library Reference*.

System information	**pidin** in the *Utilities Reference*; Analyzing Your System with Kernel Tracing in the IDE *User's Guide*; Fine-Tuning Your System in the QNX Neutrino *User's Guide*.
Target agent (**qconn**)	**qconn** in the *Utilities Reference*; IDE Concepts in the IDE *User's Guide*.
Threads	*pthread_** and *Thread** in the QNX Neutrino *Library Reference*; Processes and Threads in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*; Programming Overview in the QNX Neutrino *Programmer's Guide*.
Time	Configuring Your Environment in the QNX Neutrino *User's Guide*; *Clock**, *time()*, and related functions via their "See also" sections in the QNX Neutrino *Library Reference*.
Timestamps (for files)	**touch** in the *Utilities Reference*; *utime()* and related functions via its "See also" section in the QNX Neutrino *Library Reference*.
Timers	*alarm()*, *Timer**, and *timer_** in the QNX Neutrino *Library Reference*; Clocks, Timers, and Getting a Kick Every So Often in *Getting Started with QNX Neutrino: A Guide for Realtime Programmers*.
Transparent distributed processing	Using Qnet for Transparent Distributed Processing in the QNX Neutrino *User's Guide*; Transparent Distributed Processing via Qnet in the QNX Neutrino *Programmer's Guide*; **lsm-qnet.so** in the *Utilities Reference*; *netmgr_** in the QNX Neutrino *Library Reference*.
USB	Connecting Hardware in the QNX Neutrino *User's Guide*; **devu-***, **devh-usb.so**, **devi-hid**, and **io-usb** in the *Utilities Reference*; *USB DDK*.
Unicode	*wc*()* (wide-character functions) in the QNX Neutrino *Library Reference*; Unicode Multilingual Support in the Photon *Programmer's Guide*.
Version control	Managing Source Code in the IDE *User's Guide*; Using CVS in the QNX Neutrino *User's Guide*; **cvs** in the *Utilities Reference*; *Version Control with Subversion* (O'Reilly & Associates).
Web browsers	Web Browser Engine *Developer's Guide*

Web server (**`slinger`**)

Setting Up an Embedded Web Server in the QNX Neutrino *User's Guide*; **`slinger`** in the *Utilities Reference*.

XIP (execute in place)

Building OS and Flash Images in the IDE *User's Guide*; Writing an IPL Program in *Building Embedded Systems*.

Related reading

On QNX Neutrino

- Krten, Robert. 2003. *The QNX Cookbook: Recipes for Programmers*. Ottawa, ON, Canada: PARSE Software Devices. ISBN 0-9682501-2-2.

  For more information about this book, see **`http://www.krten.com`**.

On POSIX

The latest POSIX standards documents are available online here:

`http://www.opengroup.org/onlinepubs/007904975/nframe.html`

For an up-to-date status of the many POSIX drafts/standards documents, see the PASC (Portable Applications Standards Committee of the IEEE Computer Society) report at **`http://pasc.opengroup.org/standing/sd11.html`**.

In addition to the POSIX standards themselves, you might find the following books useful:

- Butenhof, David R. 1997. *Programming with POSIX Threads*. Reading, MA: Addison-Wesley. ISBN 0-201-63392-2.
- Gallmeister, Bill O. 1995. *POSIX.4: Programming for the Real World*. Sebastopol, CA: O'Reilly & Associates. ISBN 1-56592-074-0.

On TCP/IP

- Hunt, Craig. 2002. *TCP/IP Network Administration*. Sebastopol, CA: O'Reilly & Associates. ISBN 0-596-00297-1.
- Stevens, W. R., Fenner, B., Rudoff, A. 2003. *UNIX Network Programming, Volume 1: The Sockets Networking API*. Third edition. Reading, MA: Addison-Wesley Professional. ISBN 0-131-41155-1.
- Stevens, W. R. 1998. *UNIX Network Programming, Volume 2: Interprocess Communications*. Second edition. Upper Saddle River, NJ: Prentice Hall. ISBN 0-130-81081-9.
- ———. 1993. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA: Addison-Wesley. ISBN 0-201-63346-9.

- ———. 1995. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, MA: Addison-Wesley. ISBN 0-201-63354-X.

Some of the advanced API features mentioned in these TCP/IP books might not be supported.

Getting quick help with the `use` command

Similar to the UNIX **man** command, Neutrino presents a simple usage message for each command-line utility. At the OS system prompt, type:

use *utility_name*

and you'll see a brief description as well as the command-line options for that utility.

Hover help in the IDE

When working on your projects in the IDE's editor, whenever you hover your mouse pointer over a function name in a line of code, you'll see a popup text box containing that function's purpose and synopsis (provided that it's a documented Neutrino function).

For details, see the chapter on C/C++ development in the IDE *User's Guide*.

Visit `http://www.qnx.com`

The QNX Software Systems website (**http://www.qnx.com**) offers help on using the QNX Software Development Platform through such facilities as:

- detailed lists of supported hardware
- technical articles, white papers, web seminars, and other resources
- Licensing area, which gives details on product licensing
- Partners area, which includes lists of distributors and resellers around the world
- information about technical support

Our Foundry27 community website (**http://community.qnx.com**) includes:

- source code for the kernel, core networking, and more

Some source code is available only to customers and partners with an approved QNX Restricted Content Application. To apply to see this code, download the Restricted Content Application Form from our website, fill it in, and then submit it to `licensing@qnx.com`. For more information, see the Updated QNX Source Access Policy FAQ on Foundry27.

- BSPs (Board Support Packages)
- technical forums — members of the Neutrino user community at large as well as our own developers and staff frequent these forums daily
- and more

For general help with Eclipse, the open platform for our IDE, visit the Eclipse consortium website (`http://www.eclipse.org`). There you'll find valuable support in the form of newsgroups, mailing lists, articles, and more.

Support plans

You can access a wide range of support resources, depending on the particular support plan that you purchase.

For more information about our technical support offerings, including email addresses and telephone numbers, please see the Support section of our website (`http://www.qnx.com`).

For more information about setting up your myQNX account and registering your products and support plans, see *Accessing Online Technical Support*. There's a printed copy in the QNX Software Development Platform box, and a PDF version on the DVD and on our website.

Training

QNX training services offers many hands-on courses at your choice of location (QNX headquarters, various training centers around the world, or your site). For details, including the current training schedule, see the Training area in the Services section of our website (`http://www.qnx.com`).

Custom engineering and consulting

Depending on the nature of your particular project, you may choose to engage us to help in areas such as:

- custom BSPs
- device driver development
- hardware troubleshooting and integration

- application development
- migration/porting services

For more information, see contact your local sales representative.

Glossary

Aviage

Product name for middleware developed by QNX Software Systems.

BSP

Board Support Package — a set of software components (**IPL**, **startup code**, drivers, **buildfile**s, as well as the **QNX Neutrino RTOS**) and documentation intended to help you get Neutrino up and running on a particular board.

buildfile

A “control” file that specifies the particular startup program, environment variables, drivers, host-target communications, etc. that will be used to generate an OS image. The file provides instructions to a utility program such as **`mkifs`**, which generates OS images. In the context of the **IDE**, the buildfile is sometimes called a **target file**.

Development License

Development using QNX Software Systems products is governed by the terms of the applicable developer license agreement to which you you must agree to install the software product. For more information, see the Licensing area of our website, **`http://www.qnx.com/`**.

Distribution License

Distribution of QNX Software Systems products is governed by the terms of the QNX Runtime License Agreement or the QNX OEM License Agreement. For more information, see the Licensing area of our website, **`http://www.qnx.com/`**.

DDK

Driver Development Kit — a set of source code and documentation intended to help you write your own drivers for various devices: audio, graphics, input (mice, keyboards, etc.), network, and USB.

Eclipse

Name of a tools project and platform developed by an open consortium of vendors (Eclipse.org), including QNX Software Systems.

The IDE within the QNX Momentics Tool Suite consists of a set of special plugins integrated into the standard Eclipse framework.

EULA

End User License Agreement.

IDE

Integrated Development Environment — in QNX Momentics, the IDE is based on the Eclipse framework, and includes many highly integrated tools for code development and system analysis.

IDE also stands for Integrated Drive Electronics, a standard interface used between a computer's bus and disk storage devices.

IPL

Initial Program Loader — the software component responsible for setting up the machine into a usable state, such that the **startup code** can then perform further initializations.

Momentics

Product name for the tool suite created by QNX Software Systems for its Neutrino RTOS.

Neutrino

Product name of the RTOS created by QNX Software Systems.

OCL

Open Community License.

PPS

Persistent Publish/Subscribe, a resource manager that makes it easy to disseminate information to interested processes.

QCL

QNX Community License.

QNX

Name of an earlier-generation RTOS created by QNX Software Systems. Also, short form of the company's name.

QoS

Quality of Service — a policy (e.g. `loadbalance`) used to connect nodes in a network in order to ensure highly dependable transmission. QoS is an issue that often arises in high-availability (**HA**) networks as well as realtime control systems.

QSS

QNX Software Systems.

RTOS

Realtime operating system.

Runtime License

See the Licensing area of our website, `http://www.qnx.com/`.

startup code

The software component that gains control after the IPL code has performed the minimum necessary amount of initialization. After gathering information about the system, the startup code transfers control to the OS.

Index