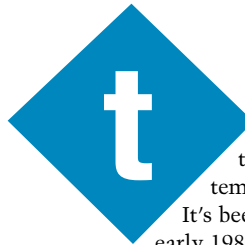


Exploring QNX Neutrino

Duane Mattern

Yes, QNX has been around for a while. But, that doesn't mean it's outdated nor that it isn't a helpful solution to some of today's real-time problems. Duane takes us step by step through everything from downloading the eval copy to displaying bootable images in real time.



The QNX real-time operating system (RTOS) isn't new. It's been around since the early 1980s. When I first encountered QNX in '87 I thought, "this is a lot better than Windows." Today, it's still better than Windows for real-time applications. In addition, QNX has added a lot of GUI applications, making it competitive on the desktop.

Since September 2000, we've been able to download QNX real-time platform (RTP) for evaluation, prototyping, personal use, or other noncommercial purposes at no charge. And, since July 2001, RTP has included the x86 version of QNX Neutrino 6.1. Throughout the article, I will refer to these as QNX unless a distinction needs to be made.

What is QNX? QNX is a RTOS with a microkernel architecture. It supports MIPS, PowerPC, SH4, StrongArm, and x86 hardware. It is scalable from constrained embedded to multiprocessor platforms. The architecture provides multitasking, priority-driven preemptive scheduling, synchronization, and TCP/IP protocol. Utilities including PPP, DHCP, NFS, RPC, and SNMP are provided as well.

QNX has native message-based networking called Qnet. The Photon microGUI windowing system is a GUI with a small memory footprint. For GUI applications, there is also an integrated development environment called Photon Application Builder (PhAB), which reminds me of Visual Basic with drag-and-drop controls. And, don't forget about the self-hosting capabilities that

simplify development. QNX is not Unix but they have a lot in common. QNX even uses a version of the GNU GCC compiler.

For anyone who wants to do hard real-time work, QNX is a good place to start. If you've ever attempted to do real-time work under Windows, you know how difficult it can be to get deterministic timing. Windows CE is a possibility if you want to drop \$2500 (US \$) in a hurry for Platform Builder. For simple jobs, you can jump backward to MS-DOS, but then you start to have problems with software and hardware support.

Linux is a possibility for real-time applications, but you have to make a tough choice. Which real-time implementation of Linux do you want? There are many choices to evaluate. The list includes RTLinux from FSMLabs, Embedix from Lineo, Hard Hat from MontaVista, TimeSys, and others. [1] The different approaches to scheduling and the supported hardware complicate the choice. If you want to start quickly and for free with a proven RTOS, then QNX is a good candidate to explore.

INSTALLATION

Let's get started by exploring the x86 installation. I've installed QNX on generic 1997 PC hardware, a Dell Dimension 4100, Dell Inspiron 7500 notebook, as well as the PC/104 platform (Panther/K6 from VersaLogic (see Photo 1)). As with any installation, there are quirks. Here are some recommendations if you want to get started quickly.

First, get a second x86 platform and install QNX as the only OS on that platform. With a Pentium processor, PCI bus,

Main menu	Submenu
Editors	Notepad, jed, Vim
Utilities	File Manager, Image Viewer, Snapshot, phcalc, terminal, ica manager
MultiMedia	Mixer, MediaPlayer, Real Player G2, Real Player Readme
Internet	Voyager, Vmail, Phirc, Tin
Development	PhAB, DDD Debugger
Games	Columns, Doom shareware, Peg, Othello, Solitaire, Video Poker, Quake3 Arena
Software	Package Manager, Install Software from QNX, Manage My Software
Configure	Networking, Appearance, Video Display, Shelves, Font Server, ScreenSaver, Active Print Jobs, Audio Level, Input Cfg
Help	Starts Voyager web browser with local help in HTML
Quick Guide	Brings up welcome screen
Shutdown	Ends Photon session, shuts down system, and reboots

Table 1—These QNX applications are available from the shelf in the Photon GUI.

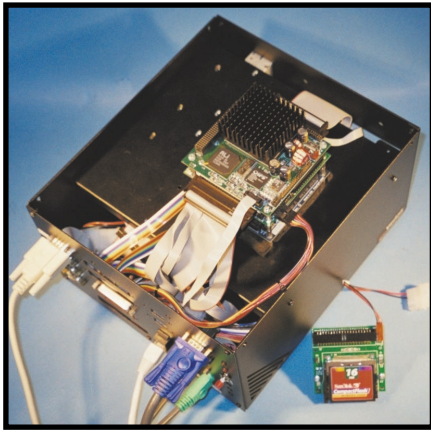


Photo 1—This is the Panther/K6, low-power (no cooling fan) board on a PC/104 stack with a Sealevel serial card, all housed in the VersaLogic development box. Note the IDE-to-CompactFlash adapter in the foreground.

PS2 mouse, IDE hard disk, and 32-MB RAM, you shouldn't have any problems with the installation. Use a keyboard/video/mouse (KVM) switch to avoid having to give up desktop space. If your old hardware isn't supported by QNX, look for supported hardware on eBay. I picked up a fully supported VGA card for \$20 on eBay.

If you don't want to use a second box, I suggest installing QNX into a Windows partition with a FAT file system. This way, you won't have to work those ugly issues with the boot sector and will avoid the possibility of corrupting your system. Sure, you'll have to use a boot floppy to boot to QNX. But the floppy isn't required after the OS is running, so you won't suffer performance problems because of floppy I/O. On my Dell Dimension, I use the Windows boot loader to select between Win2k and Linux. If I want to boot QNX, I use the QNX boot floppy.

My third recommendation is to ante up the \$30 for the QNX CD. Yes, you can save a few bucks by downloading the 29-MB Windows installation, but it includes only the basic operating system. It does not include all of the packages, so you'll have to install the packages over the Internet. The download for the CDISO image is 442 MB. It

is much easier and quicker to do the installation with the QNX CD.

It's always good advice to make a QNX boot floppy to use for recovering from problems that might occur after making changes to your system.

I have one last suggestion. I don't like to boot directly into the Photon GUI because it becomes difficult to maneuver in the GUI if you have a problem with your mouse (like I did with a Logitech bus mouse). Fortunately, there are keyboard shortcuts you can use. The "Windows" key pulls up the menu, and then you can use the cursor keys to select items from the menu. From a window, you can use the tab, space, and return keys to select the various menu items. To avoid this, I prefer to boot into a terminal interface and then start Photon with the `ph` command.

You can install QNX to its own partition, but you'll have to work through the boot sector issues. The problems you may have depend on six conditions: which operating system(s) you have, the order in which it was installed, which boot loader you want to use, if you have more than one hard disk, the size of your hard disk, and the age of your BIOS.

There isn't room here to cover all of the possible combinations (Win9x/ NT/2k, Linux, QNX, etc.). A newsgroup archive site (such as groups.google.com) can be a good place to search for this information. For example, go to comp.os.qnx and search for "dual boot." If you decide to install QNX on its own partition, you'll have to boot your system from the CD.

If your BIOS doesn't support booting from a CD, you may use the Make Floppy menu item displayed in Photo 2. After you boot QNX, you may complete the installation from the CD to a separate hard disk partition. If you have problems with the installation, try pressing the escape key at the beginning of the installation when prompted. This option uses a boot image that does not use DMA and may solve some installation problems.

To avoid discussing issues with boot



Photo 2—You can install QNX Neutrino from Windows to a Windows FAT partition. The installation window also gives an option to make boot floppy disks, should you want to install the system on a QNX partition and your system does not support booting from a CD-ROM.

installing to a FAT file system and booting from a floppy. If you proceed with the installation to the FAT32 partition, you'll be prompted to insert a floppy disk.

You may reboot into QNX after the installation is complete. After booting, QNX will locate the installation on hard disk. QNX mounts a QNX file system that is located in a file on the FAT file system. This whole procedure is fully explained in the `readme.txt` file on the QNX Neutrino CD.

After booting into QNX, you will see a log on screen within the Photon GUI. Next, a welcome screen appears, which will provide an introduction and overview of the system (upper left-hand corner of Photo 3a). Photo 3a shows the Photon GUI with several windows open, including the Voyager web browser, calculator, and a terminal window.

One of the first things to do is configure your network. You can use the network configuration tool from the shelf. The shelf is the menu shown in the enlarged view in Photo 3b. The network configuration tool is a front-end to `phlip`, the Photon TCP/IP, and dial-up configuration utility.

It should start with Devices, Dial-ups, and Network tabs (see Photo 4). If you don't have a Devices tab, it means that the network driver did not automatically start.

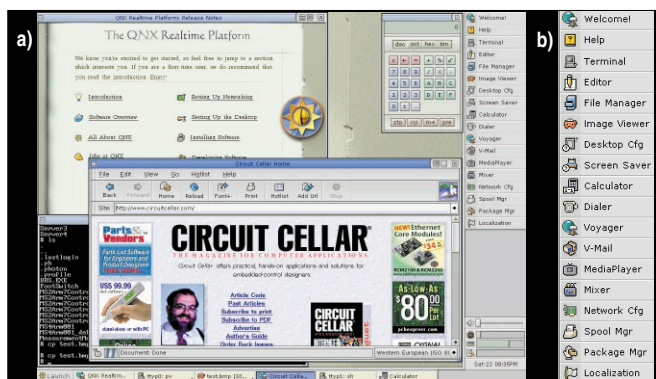


Photo 3a—After you have booted QNX Neutrino and started the Photon GUI, you'll see a number of familiar applications, including a web browser (Voyager), calculator, console, and online help. **Photo 3b**—The Launch push button in the lower left corner provides access to a menu of commands (called the shelf). From the shelf, you can configure most application settings.

loaders, let's explore the installation of QNX to a Windows FAT partition. After inserting the QNX CD under Windows, the procedure will start if the CD auto-run feature is enabled. The installation procedure detects which Windows OS you are using. If you're running NT or Win2k, the installation informs you about

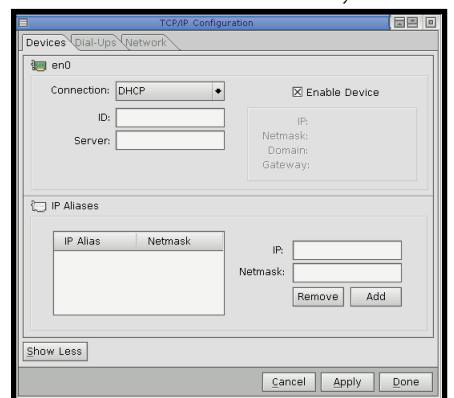


Photo 4—From the shelf, you can configure the network settings. If there is a Devices tab, it means that the system has automatically recognized your network card during the boot process. If this tab is missing, you'll have to manually start the driver for your network card, assuming that it is supported.

10-ms period		1-ms period	
Time from start	Delta time	Time from start	Delta time
0.00000000	0.00000000	0.00000000	0.00000000
0.00999847	0.00999847	0.000999847	0.000999847
0.01999694	0.00999847	0.001999694	0.000999847
0.02999541	0.00999847	0.002999541	0.000999847
0.03999388	0.00999847	0.003999388	0.000999847
0.04999235	0.00999847	0.004999235	0.000999847
0.05999082	0.00999847	0.005999082	0.000999847
0.06998929	0.00999847	0.006998929	0.000999847
0.07998776	0.00999847	0.007998776	0.000999847
0.08998623	0.00999847	0.008998623	0.000999847
0.09998470	0.00999847	0.009998470	0.000999847
0.10998317	0.00999847	0.010998317	0.000999847

Table 2—Running the `timer.c` program produces these results. The program measures the resolution of the default system timer in QNX Neutrino. The resolution is approximately 1 ms.

But, this isn't proof that your hardware isn't supported, rather, it probably means QNX couldn't do it automatically. You'll have to troubleshoot the network start up.

The debug process is outlined in the QNX welcome screen, under "Setting Up Networking" and "Quick Start—Direct Internet Connection." The latter brings up an HTML page in the browser and provides a "Network Troubleshooting" section. You can see if the driver is running in a terminal window by piping the output from the process status command (`ps`) to `grep` and searching for `io-net` as follows:

```
ps | grep io-net
```

You can use `nettrap` to detect network cards and start the driver, but if `nettrap` didn't start the driver originally, you'll have to do some manual work. Compare your network driver chipset to the supported chipsets displayed on the QNX web site. When you find an appropriate driver, you can force the system to use that driver using `io-net`.

In any event, the network configuration tool allows you to assign a fixed IP address or have one assigned using DHCP. At this point, you should have a functional desktop machine. Likely, next you'll want to invoke the package manager to install the various other utilities and tools, like a C and C++ compiler.

TOOLS

You won't be able to use the C compiler until you install it from the package. Follow the options under the package manager for straightforward directions. You can

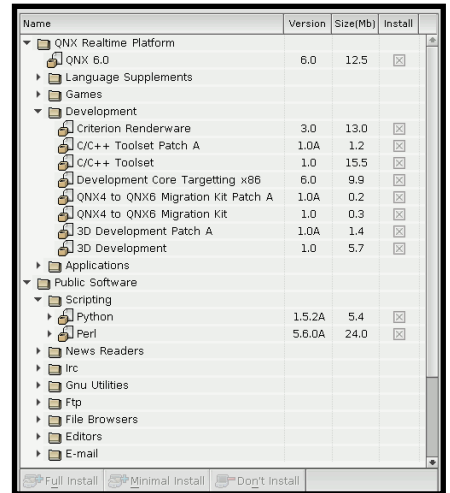


Photo 5—From the shelf, you can also run the Package Manager. The Package Manager allows you to install additional programs and applications from the QNX web site or, as shown, from the CD-ROM. This step has to be performed before a C compiler is available. What you see here is running from a QNX CD from January of last year.

connect to the software repository on the QNX web site or on the QNX CD, as illustrated in Photo 5. The packages include Korean, Japanese, as well as Chinese language supplements. QNX RTP supports C and C++ with the GCC-2.95. There is a `qcc` command, however, `qcc` is a front-end to GCC.

The purpose of `qcc` is to provide an abstraction layer to mask the differences between various compilers (GNU GCC, Watcom, and Metrowerks). There is also `cc` front-end, which is used to mask the differences between available compilers. This front-end also invokes the appropriate compiler (C or C++) based on file extensions and currently it invokes GCC/G++.

For project code management, the `make`, `CVS`, and `RCS` utilities are part of this QNX distribution. If you prefer to do cross development, there is a cross-compiler version of the GCC compiler that will allow development on Windows for a Neutrino target. But, I haven't used it, because self-hosting `make` more sense on x86 platforms.

Besides compilers, let's see what else you get. In the lower left-hand corner of Photo 3a, there is a Launch button that is similar to the Start button in Windows. This button brings up a number of submenus. The default submenus are shown in Table 1.

Snapshot (from the Utilities window) is the screen/window capture utility I used to capture images for this article. Under the Development menu, there is an option for PhAB, which is the Photon Application Builder shown in Photo 6. This is a neat utility that is similar to Visual Basic, except that it is C-based. PhAB allows you to drag-and-drop various controls to a desktop application.

You have to complete the code stubs provided by PhAppBuilder. In my experi-

```
Listing 1—This code is used to change the priority of a thread and create and configure a timer.

//Set the priority on the main thread to 28. 10 is normal.
threadMain=pthread_self();
iret = pthread_getschedparam( threadMain, &policy, &SchParam);
SchParam.sched_priority = 28;
iret = pthread_setschedparam( threadMain, policy, &SchParam);
*****
A signal is set up for the timer trigger and a callback function pointer is declared
*****
signal(SIGRTMIN,TimerCallback); //set own handler
SIGEV_SIGNAL_INIT(&event,SIGRTMIN); //MACRO
*****
A periodic timer is created, configured, and started with a 10 ms period
*****
iret = timer_create(CLOCK_REALTIME, &event, &timerid);
//Setup the clock update rate using the itimerspec structure
itimer.it_value.tv_sec = 0;
itimer.it_value.tv_nsec = 500000000; //0.5 seconds
itimer.it_interval.tv_nsec = 010000000; //10 ms
itimer.it_interval.tv_sec = 0;
timer_settime(timerid, 0, &itimer, NULL); //Start timer
```

```
Listing 2—Here is a complete C program for starting the watchdog timer on the VersaLogic Platform. Assuming that the watchdog timer has been enabled in the BIOS, running this application will cause the system to reboot following a watchdog time-out.

#include <sys/neutrino.h> //ThreadCtl()
#include <hw/inout.h> //out8()
#define PORT_WATCHDOG 0x0E0
int main()
{
    unsigned char byteFromPort =0;
    unsigned char byteToPort =0;
    ThreadCtl(_NTO_TCTL_IO,0); //writes to io-ops
    byteFromPort = in8(PORT_WATCHDOG); //in al,E0h
    byteToPort = byteFromPort | 01; //or al,01h
    out8(PORT_WATCHDOG,byteToPort); //out E0h, al
    //Kick once to start it
    byteToPort = 0x5A; //mov al, 5Ah
    out8(PORT_WATCHDOG+1,byteToPort); //out E1h, al
    return(0); //Without subsequent writes to the watchdog
} //The system will restart in 250 ms.
```

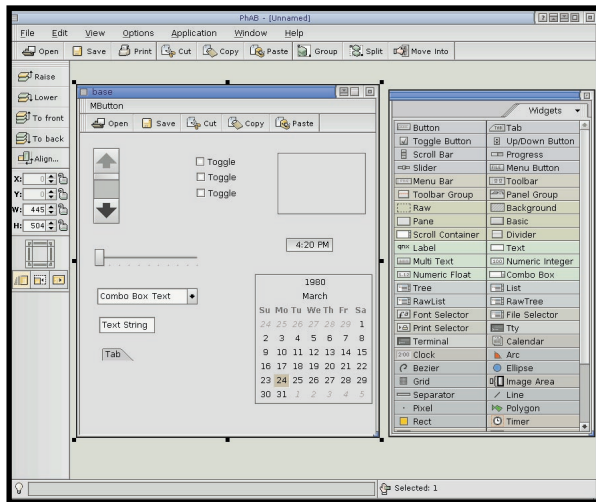


Photo 6—One of the applications available from the shelf is the Photon Application Builder (PhAB). This is an IDE for programming the Photon GUI. It works similar to Visual Basic by generating C code stubs. You can see scroll bars, push buttons, check boxes, a calendar, and more.

ence, it works reasonably well, but there are caveats. Make sure you read the documentation before starting a project, because certain restrictions are imposed to prevent PhAB from wiping out your work. Also, not all of the controls are fully functional. I was never able to get horizontal charting control (PtTrend) to function properly.

As with any package, it takes a while to learn the nuances before becoming proficient, but PhAB is a quick way to get up to speed on developing applications for the Photon GUI. Also, if you get serious about QNX, there is a wealth of third-party tools. Check the QNX web site for purchasing information.

HARDWARE

In order to evaluate QNX Neutrino, I installed it on a PC-104 Panther platform from VersaLogic. This system has an AMD K6 processor and is configured for low-power operation (no-fan heat sink). The system requires only 5 V. It's being used for prototype development and includes Ethernet, PCI-based video and IDE controller, a 2.5" hard disk, DiskOnChip, and Compact-Flash. Panther duplicates the functionality of a PC in a two-card stack.

I also installed and ran Windows 95 and Linux on this platform. The installation was completed using a floppy disk and a temporary CD. The floppy is required because the general software BIOS of the platform does not support booting from a CD. After installation, I removed the CD from the system. The video, mouse, and keyboard are connected to a KVM switch so that I can share the monitor, mouse, and keyboard that I use with my main desktop system. The QNX system also may be accessed over the network using a Telnet session, FTP, or `fs-cifs`, which is

a QNX command that allows me to mount another file system over the network.

EXAMPLES

Most likely you'll be interested in QNX if you are doing real-time work. So, the first example shows a timer callback function in a high-priority process. The timer is set to trigger every 10 ms, and because this is just an example, the timer callback function copies the time to a buffer. The buffer is printed to the screen after 50 iterations. Code segments for the key function calls are shown in Listing 1. As you can see

demonstrated in the listing, Pthread function calls are used to change the process priority from 10 to 28.

The main thread sits in an idle task until the timer triggers, and then the timer callback function is run. The results show a consistent time, but there is a finite resolution and the timer resolution can be obtained programmatically using the `clock_getres()` function. The x86 result is 999847 ns (about 1 ms). You will have to use a different approach if you need better timing than this (see Table 2).

As a second example, the code in Listing 2 demonstrates the functionality of the watchdog timer on the Versallogic Panther platform. The watchdog timer first must be enabled in the BIOS. Next, the port must be enabled programmatically. After enabling the watchdog, the system will restart unless the watchdog timer is written to within 250 ms. When run, the example routine in Listing 2 will reboot because the watchdog timer will time-out.

As a third example, I created two bootable images and then relocated them to their prospective boot devices, one on CompactFlash and the other on DiskOnChip. There isn't enough space here to go into the details, but I included code on *Circuit Cellar's* ftp site that specifies the steps used to perform these tasks and how to make a bootable image.

WHAT'S MISSING?

One thing that would benefit new users of QNX RTP is introductory literature. With Linux or Windows you can find literally hundreds of books of information. But, you'll find only one book about QNX and it was written by Robert Krten. [2] Krten's books is good, but it isn't a beginner's books nor an introduction. So, be prepared

to dig into the provided help files, use the `man` command in the terminal window, and search the newsgroup using an archive site like groups.google.com. Use the various QNX web sites to obtain the information you need to use QNX. That's another reason for using a separate platform for QNX. You can use one platform as a document resource while making modifications to the QNX platform.

WHAT YOU GET

Determinism, real-time, POSIX compliance, self-hosting, priority-based preemptive scheduler, and a free evaluation copy. What more could you ask for? Sure, your next question will be what does it cost for commercial use? Sorry, you'll have to contact QNX and ask them directly for that information. ☒

Duane Mattern is an instrumentation and controls engineer with 10 years of experience in the areas of modeling, simulation, control system design, and implementation. You may reach him at d.mattern@ieee.org.

SOFTWARE

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2001/138/. A PDF file that provides a procedure for building a QNX bootable image and installing it is also included.

REFERENCES

- [1] Linux Devices, "The Real-Time Linux Software Quick Reference Guide," rev. January 19, 2001, <http://www.linuxdevices.com/articles/AT8073314981.html>.
- [2] R. Krten, *Getting Started with QNX Neutrino 2—A Guide for Realtime Programmers*, 2nd ed., PARSE Software Devices, Kanata, Canada, www.parse.com/books/index.html, August 2001.

RESOURCE

Server: inn.qnx.com. Newsgroups: qdn.public.qnxrtp.installation, qdn.public.qnxrtp.newuser.

SOURCES

QNX real-time platform
QNX Software Systems Ltd.
(800) 676-0566
Fax: (613) 591-3579
get.qnx.com

Panther platform
VersaLogic Corp.
(541) 485-8575
www.versallogic.com

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuitcellar.com/subscribe.htm.