QNX Momentics for Visual Studio Users

Authored by: Dennis Kelly

Purpose

Since QNX makes its Momentics package available for all to download and evaluate, this document attempts to orient the base of Windows Visual Studio users to the feature-rich Momentics IDE environment. It is intended to allow the Windows programmer to leverage their experience and become proficient with QNX Momentics.

Terminology

This document uses "Visual Studio", "Studio" and "VC++" interchangeably.

This document also relies on concepts which are common to both "Visual Studio 6" and "Visual Studio 2003/2005", without differentiation. It is assumed that the reader is familiar with at least one of these Microsoft environments.

Introduction

Writing and debugging a program using QNX Momentics has some fundamental differences from the usual Windows environment. First and foremost, when running Momentics from a Windows host…

- ***The program must be run, and/or debugged, on a target other than the development machine.***
  - When you compile a Win32 program with VC++, it is typically debugged on the PC that creates the program.
  - When using QNX Momentics from Windows, you are **never** compiling for, and running on, your local operating system.

QNX Momentics "cross-compiles" your program for a specific target processor, which will be running the QNX6 (Neutrino) operating system.

- The target ***may*** be x86, but could just as easily be for PowerPC or ARM.
- ***If*** your target is x86, then **you may use a virtual PC** as your target.
- Even if your ultimate target is not x86, you may want to use a virtual PC as a convenient way to debug complex algorithms before running them on your target.

While providing a mechanism for producing transportable code, cross-compilation requires a paradigm shift to a slightly more complex IDE experience. This "new way of thinking" is the reason for this document.

Preparing a Target

Hosting QNX Momentics on Windows means that you are ***always*** compiling for a different machine. You can't really use Momentics until you have a target you can connect with, so lets do this first.

> *The easiest solution is to use VMware (or similar product) to create a target virtual machine running QNX Neutrino.*

1) Create an empty virtual machine. This does not require a large amount of disk space; about 500M is the minimum partition required.
2) Burn the Neutrino runtime ISO to CD, and "install" from this CD into the virtual machine.
3) Enable a "private shared network" with Windows.
4) Set the number of processors to "2" if you have a dual core PC.
5) Login as "root" (no password) on the target machine.
6) Right-click and select "Terminal".
7) Launch the QNC Momentics target debug agent by typing "qconn" in the terminal window.
8) Type "ifconfig" to see the IP address assigned, or click **Configure --> Network** on the list of buttons to the right of the GUI desktop. *You will need the displayed IP address to connect from Windows*.
9) Ping from Windows to QNX6, and from QNX6 to Windows to verify IP connectivity.

Now that you have a target running, the debug agent ("qconn"), and you know its IP address, you are ready to create and debug your first program.

Creating and Building a Program in QNX Momentics

Creating a program in QNX Momentics is very similar to the process in VC++. With both application, you use a wizard to produce a skeletal project of the type you selected.

Like Studio, Momentics has a concept of '**workspace**'. And as in Studio, a workspace can contain individual '**projects**'.

One difference between Momentics and VC++ is that workspaces and projects are directory-based. In other words, a workspace is the root of a directory tree containing subdirectories (folders) named for the projects. Metadata is stored in an xml file named .project within each project directory. (With VC++ workspaces, projects need not be located in their own directories, but continue to be represented as trees in the left navigation panel.)

When you open Momentics you *must* specify the workspace you want to use (unless you have selected to use one by default). If none exists, an empty workspace will be created for you.

Once a Momentics workspace opens, the IDE will seem very familiar to Studio users.

- A left vertical panel contains a tree structure representing the projects in the workspace.
- A larger panel to the right of the tree is available for editing source.
- A bottom panel is available for messages.

Now, you can create your first project in the open workspace. Select **File --> New --> Project** from the top menu. From the wizard you want to select "QNX C Project" and type a name. For example, for the illustrations, "try1" was used.

Since Momentics will perform a cross-compile for your target, before project creation completes, you **must** select the "Build Variants" tab on the dialog. *Here you will select the target processor type you will use and select Debug and/or Release versions.* You *may* choose to build for all processor targets, but this will likely cause unnecessary delays when recompiling.

When you select "Finish" you will have created a QNX C project including source for a typical "Hello, World" program.

To illustrate some of the finer points of the IDE, two or more projects are required. So repeat the steps above and create a second project. This time select "QNX C++ Project". The second project is named "try2" for the illustrations.

A word about C and C++

You will find most software for QNX6 is written in C as opposed to C++. This is not because C++ is unsupported or considered inferior. It is mostly due to the fact that the majority of the software you will see written for QNX6 is closer to 'system' software than to 'application' software as you might see in Windows.

Traditionally, system software is written is C. This is true for QNX6, as it is also true for Linux or any other *NIX-type operating system.

If you want to write in C++ for QNX6, it is fully supported by the gcc compiler in Momentics. However, you will notice that when you make operating system calls, you will be calling C functions. *This is the same as in Windows when you make calls to the Win32 SDK*.

Note: A C++ program in Momentics has the default extension ".cc".

Like VC++, you build your project in the IDE before you debug the project. An easy way to build a project is to right-click on the project name in the left panel, and then select "Build Project". Using this method you can selectively build one project. This is equivalent to first selecting the project you wish to build in the left panel, and then selecting **Project →Build Project** from the top menu.

*The second method illustrates a point – the behavior of many of the IDE controls is sensitive to **which project is highlighted** in the tree view panel.* In other words, you select the "active project" in Momentics by selecting it in the left navigation tree. For example, the Console shown at the bottom of the C/C++ perspective changes to show the build output for the currently selected project.

In Visual Studio 6, you select the "Active Project" from a pulldown menu containing the list of all projects in the workspace. This will change the project to "highlighted" in the tree view. When you select Build (F7), it builds only the active project. (Later versions have similar project selection under the 'Configuration Manager'.)

Another way to build is to use the toolbar button  . This button builds all projects, not just the highlighted one. Note that the Console output panel *only* shows the build text for the active project. *To see the build text from another project, you will need to select that project in the tree view.*

With any type of build operation, the "Problems" tab in the bottom pane shows compile warnings and errors from ***all*** projects - not just the selected one!

Debugging a Program on Your Target

Now that your "Hello, world" programs are built, and you have a QNX Neutrino operating system installed and communicating with Windows, you are ready to begin a debug session.

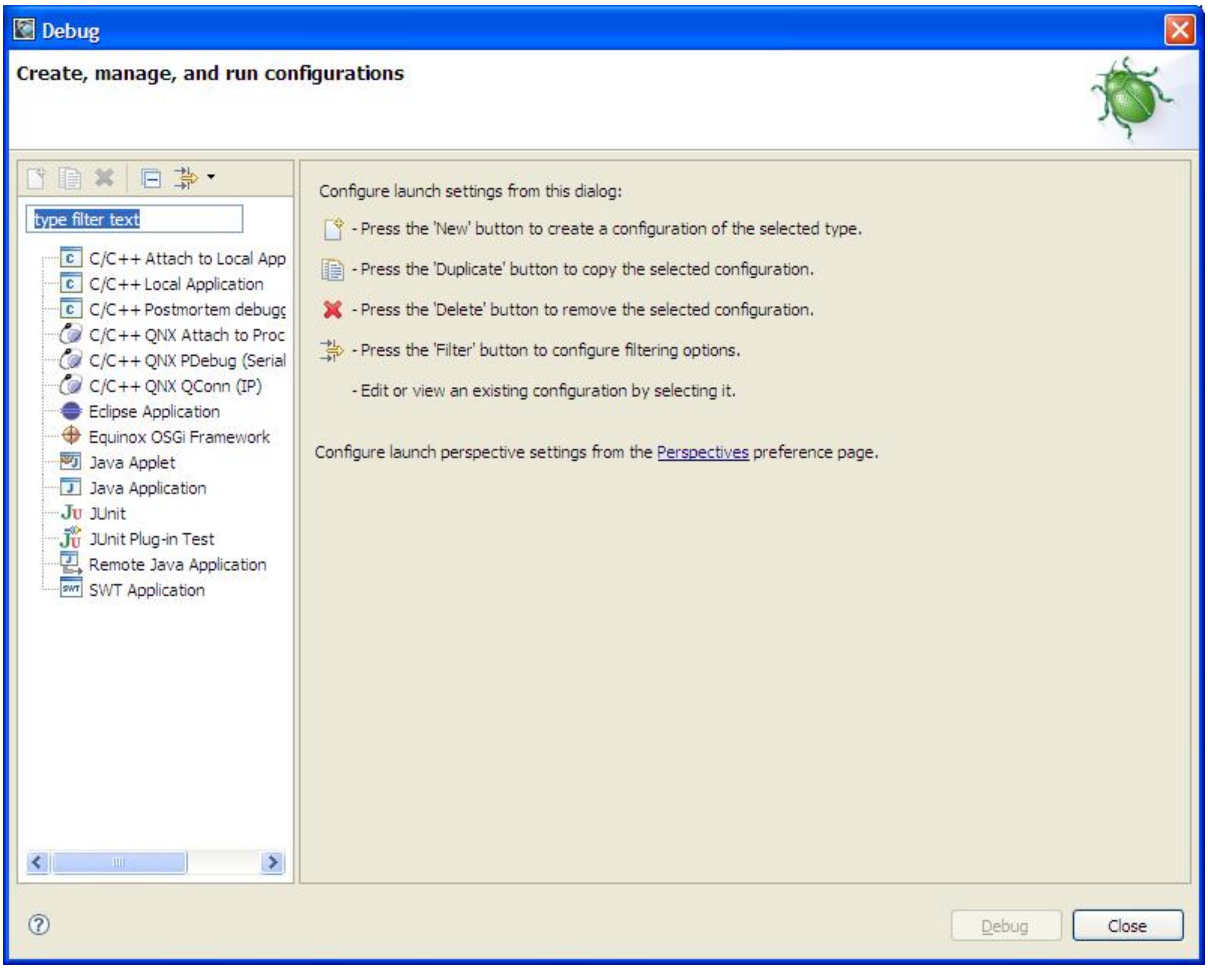
As with Studio, when you begin to debug, the workspace changes to a different configuration. In Studio, when you select "Start Debugging" the toolbars change along with the layout of the panels. When you select "Stop Debugging", the layout reverts back to its original form.

In Momentics, a similar thing happens when you begin to debug your program. In Momentics terminology, the IDE **changes perspective** from "C/C++" (the default opening perspective) to "Debug".

While in Studio you are familiar with two perspectives, in Momentics there are many. You can see a list of all of the perspectives by selecting the top menu option ***Window → Open → Perspective → Other****. Since Momentics is highly configurable, the complete list of perspectives depends on the plugins installed in the IDE.*

With Studio, to see the "debug" view you must actually begin to debug your program. In Momentics, you may select any perspective, including the Debug perspective, at any time. Typically, the uppermost tabset on the right contains (at least) "C/C++" and "Debug" perspectives.
To begin a debug session:

1. From the top menu select **Run → Debug…** or press  from the toolbar to display the following Debug window.

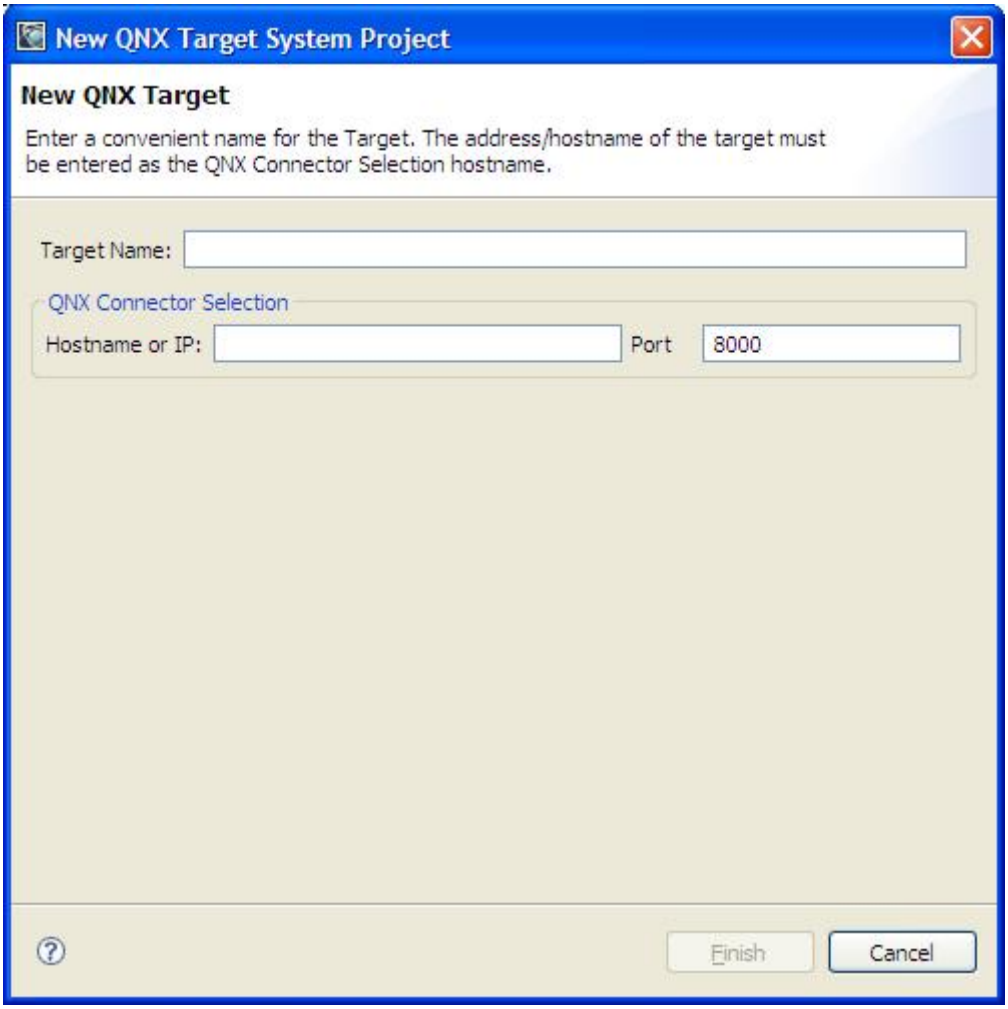


A configuration is required to build an association between a project, its executable and the target on which it will run and be debugged. Such an association is called a "Run Configuration".

2. To start, select "C/C++ QNX QConn (IP)" in the left panel.
3. Click the "New" button as indicated on the first line in the right panel.

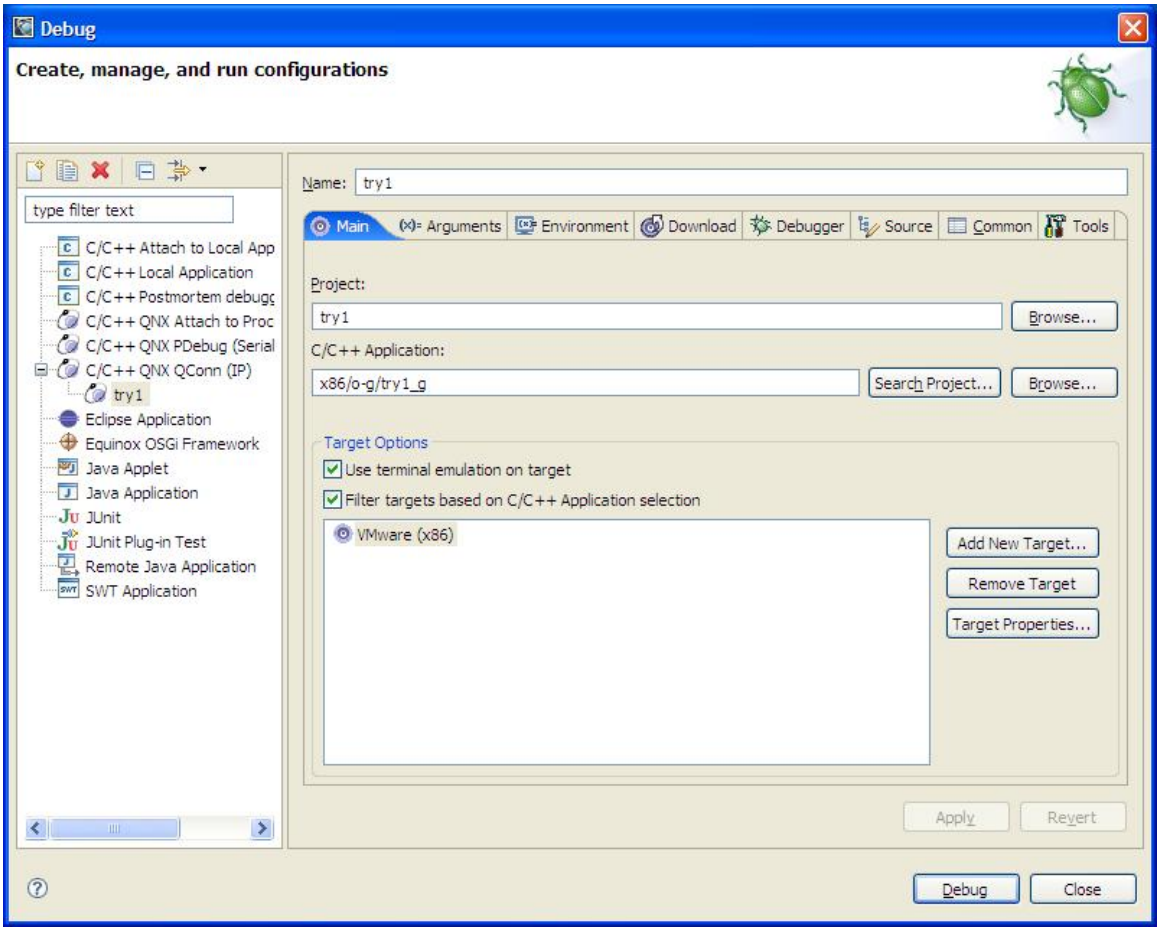
Remember that you started "qconn" on the target operating system to act as the debug agent, and it will communicate over the IP connection that exists between Windows and the virtual machine.

4. In the "Name:" field, type "try1".
5. In the "Project:" field click Browse and select "try1".
6. For the "C/C++ Application:" field, click "Search Project…" and click "OK" to use the default.
7. Now click "Add new target…" to obtain the following dialog.

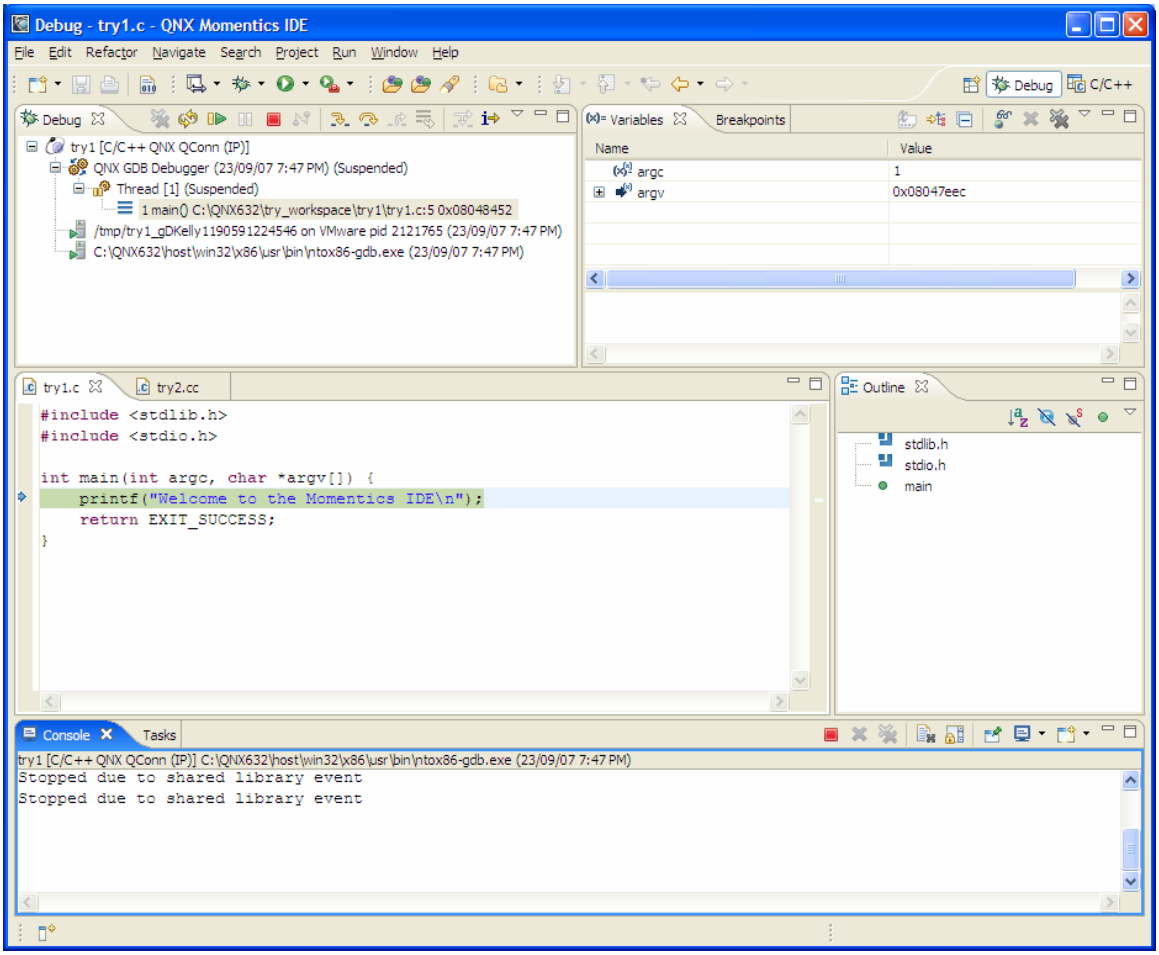


8. For “Target Name:” type something like “VMware”.
9. For “Hostname or IP:” type the IP address of the virtual machine.
10. Click “Finish”.
11. Select the target name you specified earlier (“VMware”), and click “Apply”.

The “Debug” button at the bottom is now enabled. The screen should look something like the following.



When you click "Debug", Momentics displays the Debug perspective and the first executable line of your program will be highlighted. It should look something like the following.



12. To step to the next line, click  on the toolbar. (This would typically be F10 in VC++.) If the next line were a function that you wanted to *step into* you would have clicked  . (This would typically be F11 in VC++.)

13. *To view or configure the function keys for debugging, from the top menu, select **Window → Preferences**. Then expand General/Keys. You can see that the defaults are F6 for "step over" and F5 for "step into".*

To set a breakpoint, simply double-click to the left of the source line where you would like to break.

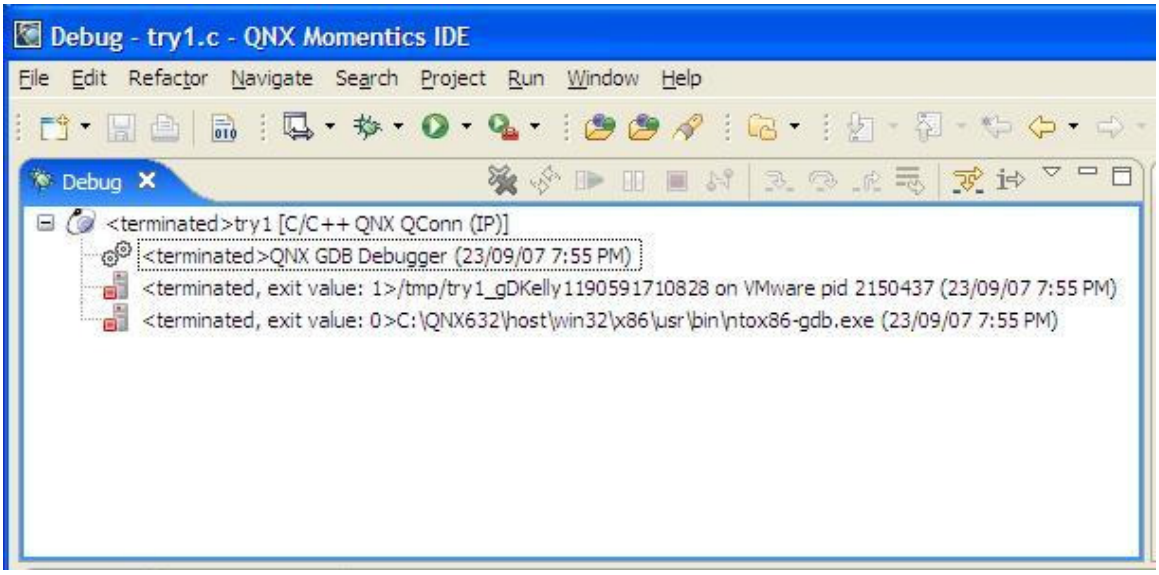
The information if the upper left panel provides the details for the debug session. Typically, you don't need the following detailed information, but it may be useful to know what components are involved:

- The executable produced by the cross-compiler was sent to the target and stored as "/tmp/try1_gDKelly1190591224546.
- The executable was started on the target with pid (process ID) 565285.
- A debug agent, "C:\QNX632\host\win32\x86\usr\bin\ntox86.exe" was started on Windows for this session to handle the communication with the target.

If you run "pidin –p 565286" on the target, replacing the number with the **pid** from your debug session, you will see your running process.

If you run Task Manager on Windows, you will see "ntox86-gdb .exe".

If you click  , the program will run until it encounters a breakpoint (or to completion if none are set). When the program terminates, the upper left panel looks something like the following.



At this point, the program on the target has run to normal completion (second line) as well as the debugger component on Windows (third line). If you click  , the window will be cleared.

Momentics is a Multi-Session Debugger

There is one more concept where the QNX Momentics IDE deviates from the Studio model.

Remember, when you begin a Studio debug session, the view changes to Debug view; the tree view of your project is no longer available. If you edit source in the Debug window

and request a compile, you usually receive a dialog informing you that, "This command will stop the debugger." (The exception occurs when the compiled change can be patched into the running executable.)

With Momentics, while you have a debug session running, you can return to the C/C++ perspective without disturbing your debug session. You could select another project and compile it. You can then start debugging the second project – *all without disturbing the original debug session.*

Multisession is useful when debugging two processes that communicate with each other, i.e. client and server. With Momentics, you can debug *both* programs in the same instance of the IDE. With Windows, you would have to start one Studio instance for the server, and a second Studio instance for the client.

If you attempt to multisession with "try1" and "try2", you will need to create a second run configuration. We only created single run configuration (for try1) above. You would need to create one for try2 as well. To do this, from the C/C++ perspective, right-click "try2" in tree view, select 'Debug As' and then 'Debug…'. The list on the Run Configuration panel will show the configuration you previously saw for try1. Change the name to 'try2'. Select 'Browse' beside Project and select try2. Select 'Search Project…' beside C/C++ Application and select the default. Highlight the virtual PC target and click Apply and Debug. You have now created a Run Configuration for try2 (as well as the one you previously created for try1).

Since you clicked Debug on the Run Configuration panel, the perspective changes to debug and try2 has been launched (suspended) on the target.

From the toolbar, press the down-arrow on the debug icon  . Select "Debug…". In the left panel you will see the two Run Configurations "try1" and "try2". Since "try2" is already running, highlight "try1" and click Debug. The upper left window will now show instances of both try1 and try2.

*The IDE has **one** set of debug controls for **both** sessions. It is important to know that the controls belong to **the currently highlighted thread!***

Select Thread [1] for the try2 tree. Press  and you will see try2 goes to the "terminated" state. Press  and the try2 instance will disappear.

Now select Thread [1] for try1. . Press  and you will see try1 goes to the "terminated" state. Press  and the try1 instance will disappear.

You are not limited to one instance of a single project. If you press  (not the down arrow beside it) it "remembers" the last Run Configuration you started, in this case try1, so an instance of try1 is created. Pressing it again, is valid and results in a second instance of try1. Each can be debugged independently from the controls by highlighting the desired instance in the tree view.

This "multisession" concept is very different from what you are used to Studio. It can lead to one problem of which you should be aware. When you debug a project, you must kill ALL instances of that project before you can rebuild the project. Simply changing to C/C++ perspective, editing a source file and rebuilding can lead to a link error:

"cc: C:/QNX632/host/win32/x86/usr/bin/ntox86-ld caught signal 1
cc: warning - couldn't unlink C:/QNX632/try_workspace/try1/x86/o-g/try1_g (Permission denied)"

This error will only resolve when you kill all running instances of the project **before** you request a build. To reiterate, this particular issue cannot happen with Studio since VC++ it is **either** in Debug mode **or** in build mode.

Note: If you receive a "couldn't unlink" error, you need to kill any running instances before you rebuild.

Advanced Momentics Topics

Selecting Window/Open Perspective/Other (or  from the Perspectives filetab), reveals that Momentics advanced capabilities are implemented as Perspectives. These include

- QNX Application Profiler
- QNX Code Coverage
- QNX Memory Analysis

This will not be discussed here, but they are good examples of the power of the the QNX Mometics IDE - advanced features can be written as plugins. It is left for you to explore the power that these plugin perspectives provide.

Conclusion

This document has attempted to show you how to debug C/C++ programs in QNX Momentics, while pointing out significant new concepts and pitfalls relative your Visual Studio experience. Hopefully presenting the information with respect to your existing mindset, made the information easier to assimilate. Good luck!