QNX IDE Projects and Build FAQ#

How can automated builds be set up with 4.5?#

1) If you are using Makefile Projects simply call make from project directory, you have to create a super makefile to build all required projects (can create a super project for that too)

2) Using of container projects. This option could be convenient for those who use a big code base. Container is a holder of various projects (among them could be other containers as well). It is the user's decision on how to combine projects into container. For example, third-party service libraries could be assembled in a single container. Each container has a set of different build configurations (this is almost the same as assigning make targets such as debug, release etc.). To build the whole workspace just launch a root container build specifying configuration. This structure should be mirrored into command line build. To provide this, insert special make file into top container's root directory The content looks like that:

```
<configuration_1>  :
    cd ../Child_Container_1 ; make <target name>
    cd ..
    ...
    cd ../Child_Project_k ; make <target name>
    ...
<configuration_n>  :
    ...
```

3) Using mksbp utility

4) Using make for QNX Projects. In this case super makefile has to contain some setup which includes setting "project path" macros. The recommended way is to create an extra file with this definition (here mylib is the project name):

- If your project and dependency live in workspace
  - PROJECT_ROOT_mylib=$(PROJECT_ROOT)/../lib1
- If your project and dependency live outside workspace
  - PROJECT_ROOT_mylib=F:/libs/mylib

(repeat this for every project)

Then add to this extra make file launch of you "root" make file or add environment variable MAKEFILE=<extra-makefile>.

If you build several IDE project you have to create an additional makefile (can be the same as one with variables) and call make in every project, i.e.

```
 PROJECT_ROOT_mylib=F:/libs/mylib
 PROJECT_ROOT_myapp=F:/libs/myapp
all:
   cd ${PROJECT_ROOT_mylib}; make all
   cd ${PROJECT_ROOT_myapp}; make all
```

How can I create Makefile Project in IDE 4.5?#

New->C Project opens a C Project Wizard Select Makefile->Empty Project Select QNX QCC on the right (important!)

QNX QCC toolchain set environment variables, such as CC, LD etc. They are automatically corrected when you switch compiler or build variant from Settings page of project properties. If you don't want these variables to be defined you can select and undefine them from the Environment tab.

What is the difference between Managed C project with QNX toolchain and QNX Project?#

Managed C project (MBS) with QNX toolchain can be used in similar way as QNX Project. They are both managed projects. QNX Project is based on qnx recursive makefiles, when you change setting of QNX Project you modifying a makefile for this project. If you change makefile manually - QNX Project would read and recognize this change (it may be limited to known build macros). Managed project stores all setting in XML and they cannot be changed outside of IDE. You can pick builder for Managed project - internal builder, or external make builder. In the second case, makefiles would be generated that represent build process for this project, but you cannot change those makefiles. Besides that, MBS has a bit more modern UI and more flexibility for project structure (you can exclude some files to be compiled for example). On the other hand QNX Project has very good build variants structure which is nice when you are building for 5 platforms at the same time.

What is recommended way to organize project structure for IDE?#

See the technical article, "Projects, Building, and the IDE".

Where does a "QNX Project" store its metadata?#

QNX Project properties are mostly stored directly in makefiles, in particular "common.mk". Additional macro and functions stored in internal makefile ".qnx_internal.mk". Settings for current active builds are stored in workspace metadata, outside of the project. If you want it to be stored with project you can check "Share all project properties" - in this case it would be stored in .cproject file inside the project folder.

Can I create projects targeting something other than the QNX OS or can I use different compiler?#

Yes, QNX IDE is based on Eclipse CDT which allow to create project for any target. You need to install appropriate toolchain on your host and you may need to create a toolchain definition from IDE. Search CDT newsgroup for this topic.

How do I change the default settings for new projects?#

To enable QNX Toolchain as preferred toolchain for Makefile and Managed projects: * Open Window->Preferences...

- C/C++->New CDT project wizards
- Select Makefile project->Empty Project
- Select QNX QCC
- Click on button "Make toolchain(s) preferred"
- Click OK.

If you are anticipating creating many new projects and they will have values different than the defaults, you can change default settings for makefile projects:

- Window->Preferences...
- C/C++->New CDT project wizards->Makefile Project
- Go through tabs and pick defaults for your projects

For Qnx Projects:

- Window->Preferences...
- QNX->New Project
- Go through tabs and pick defaults for your projects

How to create keyboard shortcut for Build?#

To fast access to Build command define a keyboard shortcut (e.g. F6).

- Windows->Preferences...
- Type Keys in filter
- Select Keys
- Check Include unbound commands
- Type build in right panel filter
- Select Build Project
- Select Binding field
- Press keystroke <F6>
- Click Apply
- Click Copy command
- Select C/C++ Editor in When combo box
- Select Binding field and press <F6>
- Click Apply again

How show line numbers in the C/C++ editor?#

- Open Window->Preferences
- Type "line" in filter
- Select Text Editors
- Check Show line numbers
- Click OK

Is it possible to configure the IDE to add a "Build Project" icon on the toolbar?#

To show build button in C/C++ perspective (or any other perspective) from bar menu select "Windows->Customize Perspective->Commands" and check "Build Action Set".

How to create different build configuration in QNX Project?#

By default you have several platform configurations, for each of them you have release and debug variant. You can create extra configurations for example to link with different libraries. Lets say you want configuration 1 link with library libMy1.so and configuration2 link with library libMy2.so. To do that first select you platform, for example x86 (In Project->Properties...->QNX C/C++ Project->Build Variants). You have "debug" and "release" inside. Add "release2" and "debug2" (by selecting "x86" and pressing Add button on the right). Click on "Advanced" button at the bottom. ON the top right select x86 and "release", in library tab pick Extra Libraries from pull-down menu and add your library libMy1.so. Same for debug. Same for release2 and debug2 but pick library libMy2.so this time. When you build you now would have 4 configurations.

Do I need to convert my project to a "Momentics style project" to use the IDE?#

The IDE wants you to narrow down the scope of what it needs to know about (source, binaries etc) so you need to create a project _associated_ with your source/binaries and this project is in turn _associated_ with a

workspace. This project doesn't have to be in the workspace, it can be anywhere you want it to be. Here are some options:

- Source lives in a project which lives in the workspace. This is the default when you create a new project or bring source into the IDE using Import from filesystem (which _can_ do a copy, but doesn' t have to) or using a version control plugin
- Source lives somewhere in the filesystem and we want to overlay a project at that location. You can achieve this by creating a project and changing the default location from the workspace to the location of the source.
- Source lives somwhere in the filesystem, but we don't want to create any metadata files in that location. In that case, create a project (in the workspace or elsewhere as in the previous scenario) that is empty. Then create a folder in that project and make the folder location 'point' to the source in the filesystem using the ">>Advanced" section of the dialog. This is like a unix symlink, but exists only in the IDE workspace.

See more on this topic in Project, Build and IDE Article.

Do I need to convert my build process to match an IDE project?#

The process of selecting a project type appears more complicated than it is. For _nearly_ every type of existing code with a build process, you will want to choose the "Makefile Project" type. Why? Because this project type just calls out to some external build program to build the source. Most of the time this is 'make', hence the name of the project, but it could be JAM, ANT, dmake, cons or any other builder.

If you have the luxury of starting your project from scratch, then the QNX Projects are interesting because they allow you to build multiple processors (more generally 'variants', including OS'es) with a single build based on the QNX recursive make framework. They won't 'port' well to other systems however. Managed Make Projects are interesting because they provide a full IDE graphical control and configuration, familiar to those coming from Visual Studio, and can really take advantage of the Eclipse frameowork (ie incremental compiles, links etc) but are still relatively immature.

... and if you are never going to run your build from the IDE (I never say never), then the choice is even simpler: Standard Make all the time just to identify the source as "C/C++" source and help identify the binary types. See more on this topic in Project, Build and IDE Article.

Nothing is showing up anywhere, this IDE thing is totally _busted_!#

Since one of the goals of the IDE is to simplify and automate work for developers, it needs to be told what to do, there are two (per project and global default) settings in particular that are important:

- The binary parser setting. This setting lets IDE tools (like the debug launcher) filter binary code from source code. When you see the 'Binary Parser' task running in the progress bar, that is the background thread that is iterating over the project content attempting to determine which files are binaries and which are not. This is what provides the (virtual) content for the binaries folder in the Project Explorer view and the content for the debug launcher file selection dialog when you select "Search" (vs Browse). If you don't see anything in these places:
  - a) The binary search has not come across anything yet and/or is not complete
  - b) The binary parser is mis-configured, so go and check it out: Should be QNX ELF

- Indexer/Parser settings. Internal parser is used to parse C/C++ code and create internal database with symbol references as well as to create internal syntax representation of the code, which is used for Syntax Highlighting, Code Formatting, Navigation, Refactoring, Code Completion, also to show Call Hierarchy and Type Hierarchy etc. If you using your own makefile there is no way (well not right now) to figure out which include paths and macros you are using for the build, and without this information parsing would be badly broken and you may not do any of these things. You have to configure your includes and macro definition in Path and Symbols setting of Project Properties->C/C++ Build settings.

- The debugger setting (Launch configuration in general). If code paths are not matching on host and the machine it was build on you need to create Path Mapping in Source tab of launch configuration . If you need environment variable set for your process including LD_LIBRARY_PATH you have to set it in Environment tab. To load symbols from libraries Debugger needs to find libraries on the **host** side. You need to set them in Shared Libraries tab of Debug tab (Shared Libraries tab of Tools for other qnx tools). Library version has to match on target and host with runtime soname for shared libraries. See more on that topic at Debugger FAQ page.

Back