Top Productivity Tips for Using Eclipse for Embedded C/C++ Developers

Garry Bleasdale, QNX Software Systems, gbleasdale@qnx.com

Andy Gryc, QNX Software Systems, agryc@qnx.com

Introduction

This paper presents a selection of Eclipse IDE tips and tricks gathered from:

- the QNX® development community: our engineers, techies and trainers
- Foundry27, the QNX Community Portal for open development, where we have an Eclipse IDE forum
- Eclipse.org forums
- public web sites and blogs that offer Eclipse-related expertise

The 27 tips described in this paper are the tips that we received from these sources and identified as most interesting and useful to developers. We present them here with the hope that they will help make you more productive when you use the Eclipse IDE.

About Eclipse

A modern embedded system may employ hundreds of software tasks, all of them sharing system resources and interacting in complex ways. This complexity can undermine reliability, for the simple reason that the more code a system contains, the greater the probability that coding errors will make their way into the field. (By some estimates, a million lines of code will ship with at least 1000 bugs, even if the code is methodically developed and tested.) Coding errors can also compromise security, since they often serve as entry points for malicious hackers.

No amount of testing can fully eliminate these bugs and security holes, as no test suite can anticipate every scenario that a complex software system may encounter. Consequently, system designers and software developers must adopt a “mission-critical mindset” and employ software architectures that can contain software errors and recover from them quickly. Just as important, developers must employ tools and debugging techniques that help maintain system integrity during the problem-solving process.

The tools can’t introduce changes that adversely or unpredictably affect system behavior, particularly if the system is actively provid-ing service to users. And once the developer has fixed any software component, the tools and underlying operating system should make it easy to upload and monitor the fixed version, again without affecting overall system behavior and availability.

Contents

Tip 1: Show Key Assist

The Eclipse IDE key assist feature opens a pop-up window with all the valid shortcut keys for the current context:

1. Enter Ctrl+Shift+L to open the pop-up window.

A useful key assistant feature is that you can repeat the action to open the Key Binding configuration window.

Figure 1: Using the show Key Assist feature

Tip 2: Key Binding

If you find that you use the same key sequences frequently, you can open the Key Binding configuration window to create a shortcut.

To open the Key Binding window, open the Key Assist window (see Tip 1), then repeat the action:

1. Enter Ctrl+Shift+L to open the Key Assist pop-up window.
2. Enter Ctrl+Shift+L a second time to open the Key Binding configuration window. See Tip 2.

The settings in the Key Binding window let you define a shortcut for your key sequence, and set the context for this shortcut. You can set exactly where the shortcut will be available; for example, you may want the shortcut to be in an editor window, but not in a target development window or an outline window.

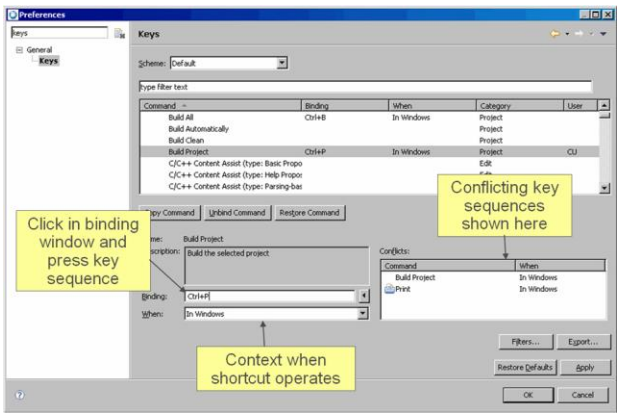

Figure 2: Binding keyboard shortcuts to commonly used functions

Tip 3: Nice-to-know Keyboard Shortcuts

The Eclipse IDE has many shortcuts, including those listed below:

Word completion	Alt+/	Multiple presses cycles through choices.
Next editor	Ctrl+F6	Brings up selection dialog.
Next perspective	Ctrl+F8	Brings up selection dialog.
Indent/unindent selection	Tab/Backtab	
Slide selection up/down	Alt+Up/Down	
Incremental find	Ctrl+J	Any navigation key ends find; use Backspace and Esc.
Maximize/restore window	Ctrl+M	
Match bracket/brace	Ctrl+Shift+P	Cursor must be just after bracket/brace (highlight showing).
Delete row	Ctrl+D	For diehards missing good ole **y** and **C-k**.

Tip 4: Refactoring

The Eclipse IDE provides simple sequences to help you refactor source code. The table below lists commonly used refactoring shortcuts:

Launch	Alt+Shift+T	Launch the refactoring menu.
Rename	Alt+Shift+R	Renames selected identifier throughout project, and warns of conflicts or shadowing.
Constant	Alt+C	Extracts and names selected constant.

IDE also provides a preview of changes, so you can step through them before committing to them.

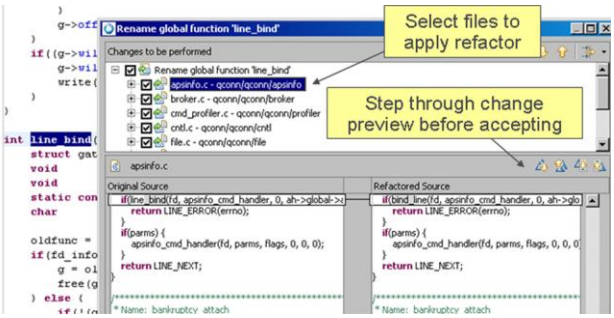

Figure 4: Selecting files to refactor

Tip 5: Call Hierarchy

Call Hierarchy shows a complete list of all functions that call the selected identifier. To use Call Hierarchy, simply:

1. Select an identifier, then enter Ctrl+Alt+H.

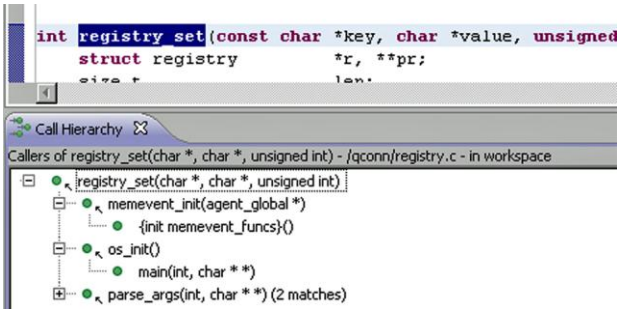

Figure 5a: Using Call Hierarchy to view a complete list of callers for a selected function or member

For example, if you are going to change a function prototype to add a new parameter to that function, you can use Call Hierarchy to see all of the functions that will be affected by your change.

View all entities called by a function

Call Hierarchy also lets you see all entities called by a selected function. To see all called entities:

1. Click a hierarchy tree button to switch the call hierarchy tree.

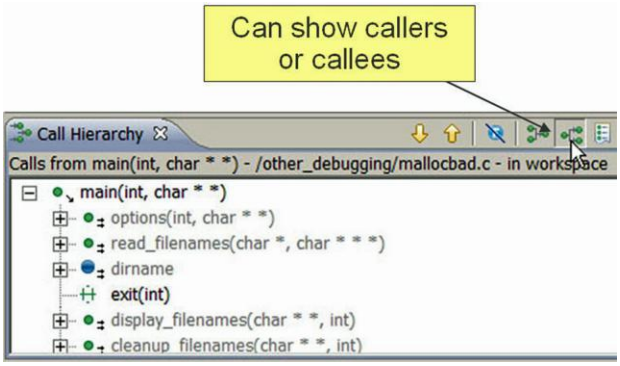

Figure 5b: Switching the call hierarchy tree to show entities calling or called by a function

This feature lets you quickly identify points at which a selected function interacts with another service or an OS function, or calls into a subroutine library.

Tip 6: Hyperlink Navigation

The Eclipse IDE provides hyperlinks to definitions and prototypes for identifiers.

```
int parse_args(int argc, char *argv[]) {
    int             c;
    int             err = 0;


    while((c = getopt(argc, argv, "p:d:")) != -1) {
        switch(c) {
        case 'd':
            if(registry_set("debug_device", optarg, 0) == -1) {
                return -1;
            }
            break;
        default:
            err = 1;
            break;
        }
    }
```

Figure 6: Using hyperlinks to view definitions and prototypes

To view definitions and prototypes:

1. Place (hover) the mouse over an identifier (function, structure, etc.) to reveal the hyperlink.
2. Click on the link to view the identifier's definition and prototype.

Tip 7: Bookmarks

Bookmarks are useful when you need to move around between different parts of a program. To use bookmarks:

1. Go to your favorite (or most infamous) places in your code.
2. Right-click on the gray, left border.
3. Select Add Bookmark.

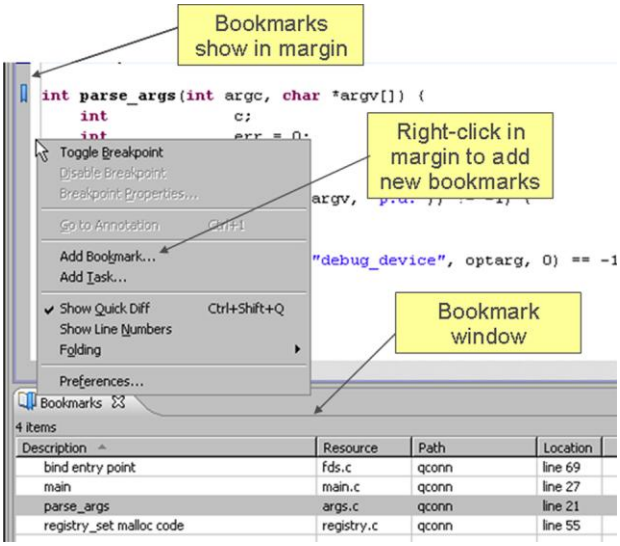

Figure 7: Making and viewing bookmarks

To view the list of books, which you can use to jump to your bookmark in the code:

1. Select Window > Show View > Other … > General > Bookmarks.

Tip 8: Prompt for Arguments During Launch

If you run code with a command-line interface that requires arguments, you may need to enter different arguments each time you run this code. Instead of creating a large number of debug launch configurations, which you will continuously have to edit, you can use the Eclipse IDE to prompt you for arguments during the launch.

The Eclipse IDE lets you include a wildcard in launch configuration arguments, so that every time you run that launch configuration you are prompted to enter the remainder of the command-line.

To set launch configuration arguments:

1. In the Launch Configuration dialog window, click the Arguments tab. See Figure 4b.
2. When you are prompted to enter a launch configuration argument, type in the string, with all the parameters that you want to pass to the relevant program when it executes.

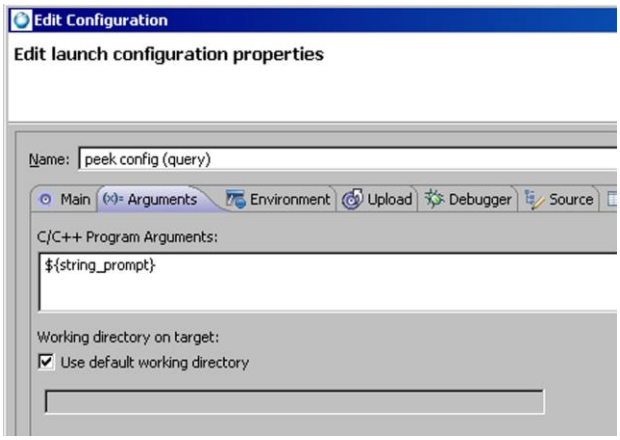

Figure 8a: Editing the launch configuration arguments

When you run the program, the string you entered becomes the default, which you can either accept or modify, as required.

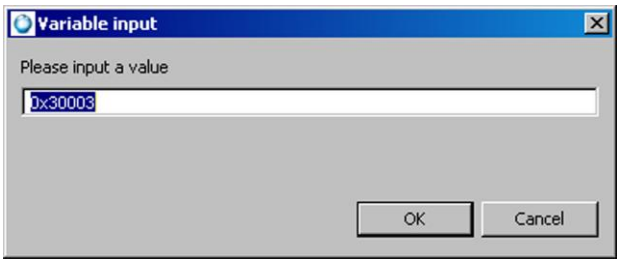

Figure 8b: Entering an argument for a launch configuration

Note that you must use the exact string for your argument: `${string_prompt}`; you can not substitute arbitrary strings for a string underscore prompt.

Tip 9: Template Proposals

Template proposals can be very useful for code, such as exception blocks, that you do not always run. If a section of code has a large number of exception blocks and you want to only fill in all braces, you can use template proposals. Template proposals are also useful for loops, because they fill in all the parts that are needed for the loops.

The IDE comes with many default templates, but you can also add your own. In addition, you can enclose the lines you have selected in the editor with a construct, such as a scope and temporary variables, in order to perform some specific operation.

To use a template:

1. Type in the first few letters of the template.
2. Press Ctrl+Space, and select the templates you want to apply.
3. Once a template has been expanded, you can enter strings in individual fields, using the Tab key to move through the fields, which the IDE fills in.

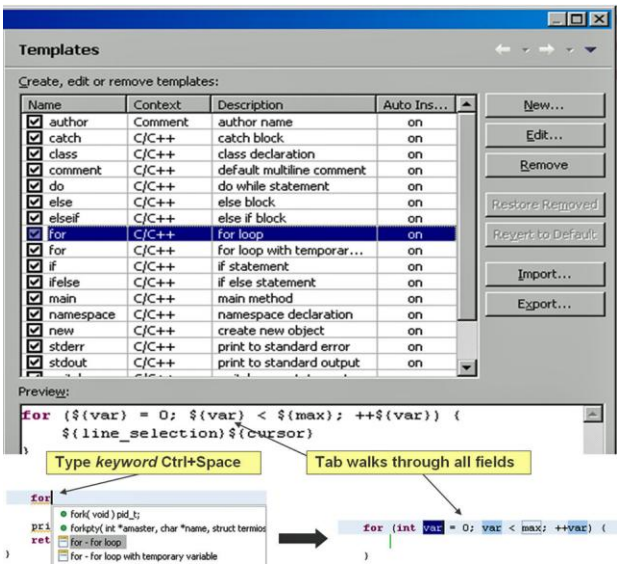

Figure 9: Applying templates

Tip 10: View Assembly Code

Many developers are not aware that they can use the Eclipse IDE to view assembly code. Though you will usually only want to open editable source code files, getting a look at the assembly code can be very useful if you are debugging with techniques such as stack traces.

If you need to examine closely a small segment of code lying in a stack frame, or if you are optimizing short code snippets, the IDE gives you a quick way to see the opcodes directly to help you understand precisely what is executing.

Once you have compiled your files and you have the object files or the executables available in the IDE's project view, just open the files to see the assembly code.

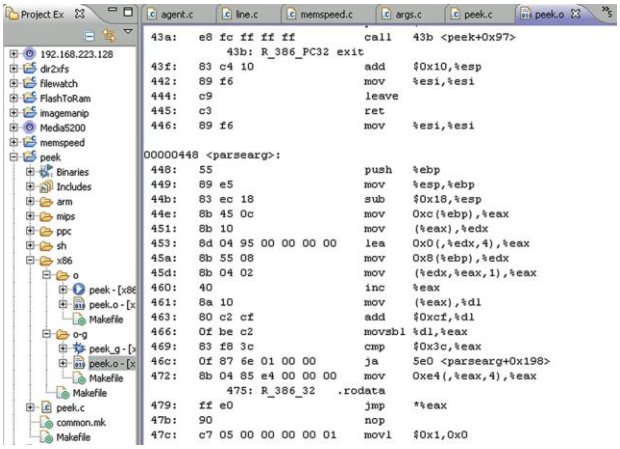

Figure 10: Viewing assembly and source code

Tip 11: Detaching Views

The Eclipse IDE allows you to detach views from the main window. This feature is particularly useful if you are using multiple monitors.

To detach a view, simply right-click on its header and select Detach. To reattach a view to the main window, either right-click on the view and select Attach, or drag the view's title bar to the location where you want it to dock.

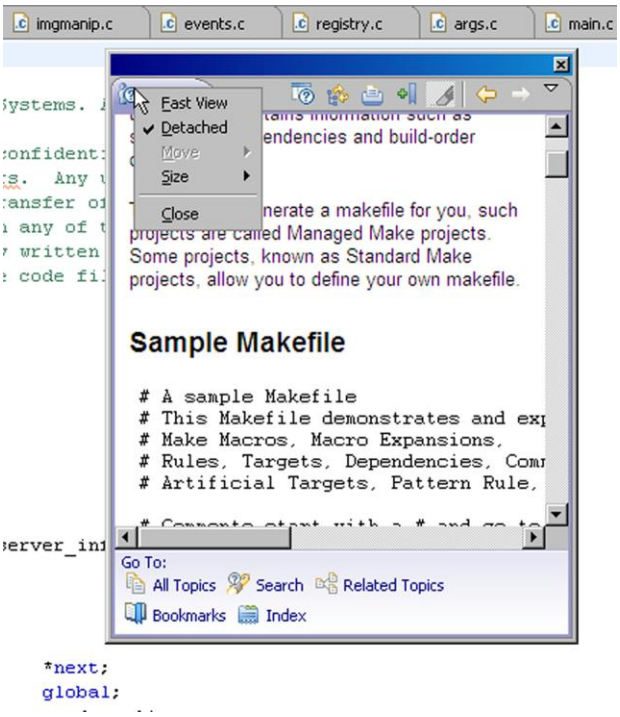

Figure 11: Attaching and detaching a view.

Tip 12: Group Launch

The Eclipse IDE can be configured to launch several processes at the same time. Multiple process launches can help debug multiple, interacting processes, such as:

- a server and its client
- a driver and its calling applications
- an HMI and supporting processes

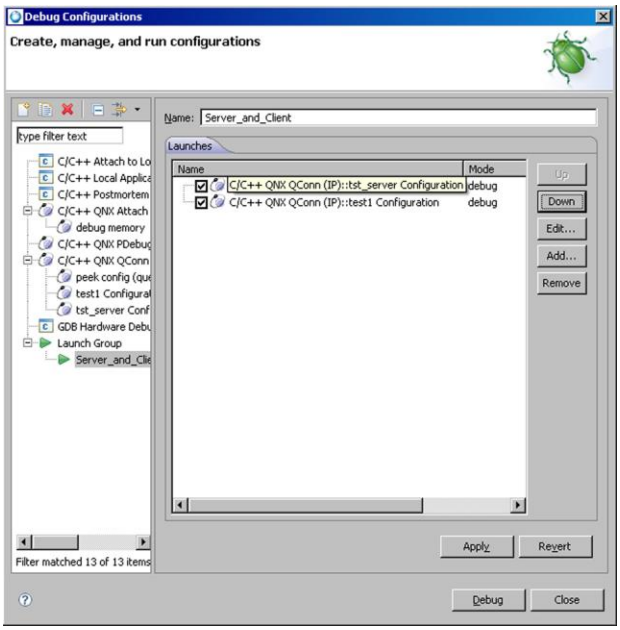

Figure 12: Using a launch group

You can combine different launch configurations, such as running a local script, kernel trace logging, or launching a remote process. If these are in a launch group, the debugger will start each member that has been paused.

Tip 13: Directory Path Variables

Directory path variables in the Eclipse IDE are similar to soft links in UNIX or Linux; they refer to a specific directory on your machine. You can use them if, for instance, the default Eclipse organization is not convenient for your build environment. A directory path variable is not restricted to a single project, so you can create variables that you will use for multiple projects.

To create a new variable for a path link:

1. Launch the New Folder dialog window.
2. Enter the link folder in the filesystem, and click the Variables button.

To see what variables have been set:

1. Select Windows > Preferences > General Workspace > Linked Resources.

You will see the variables that are already pre-configured, and you will be able to insert new variables.

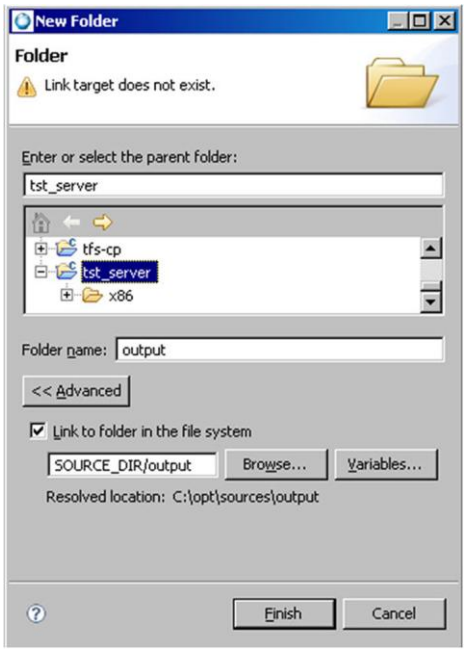

Figure 13: Viewing directory path variable configurations

Tip 14: Custom Breakpoint Actions

Custom breakpoint actions are a convenient aide for debugging code with hard to reproduce errors.

Debugging often involves running the same code repeatedly until a specific set of conditions cause it to fail. This type of debugging can mean repeating the same action dozens or even hundreds of times without error.

Custom breakpoint actions let you set up custom notifications and other actions in your debugger, so that you can leave code running and focus on other tasks until the code encounters an error in the code. You can use custom breakpoint actions to play a sound or specified WAV file that alerts you when the code you are troubleshooting reaches a breakpoint.

A breakpoint action can also have the Eclipse IDE create a log entry, or simply print (the value of a specified variable, for example), and resume. This last capability offers you a mechanism for effectively inserting `printf` statements into your code without recompiling and downloading the code.

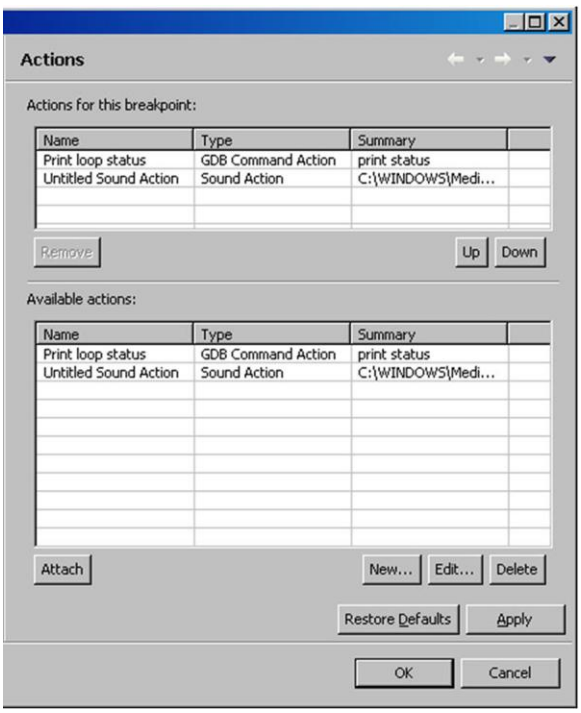

Figure 14: Setting breakpoint properties

Finally, you can set up multiple and different breakpoint options for each breakpoint. To access all these functions:

1. Right-click on a breakpoint and select Breakpoint Properties.

The Eclipse IDE will open the dialog box shown in Figure 14, which shows the options available for your custom breakpoints.

Tip 15: Manipulating Target Files

Target file manipulation is a feature specific to QNX's version of Eclipse, the QNX Momentics® Tool Suite. It is a very useful time-saving feature, and it is often the subject of questions and queries in forums.

The QNX version of the Eclipse IDE lets you manipulate files on your target. Often, developers debugging targets need to copy files to or from the target, or even edit files directly on the target.

If you have the QNX Momentics Tool Suite, the IDE has a target filesystem navigator, which you can use to explore directories and files on your target, copy files to and from your target, and perform other actions (including deleting files and launching executable files) just as though you were working on a local system. With the target filesystem navigator, you can even edit and save target files in the IDE without having to use `telnet`, `vi`, or `ftp`.

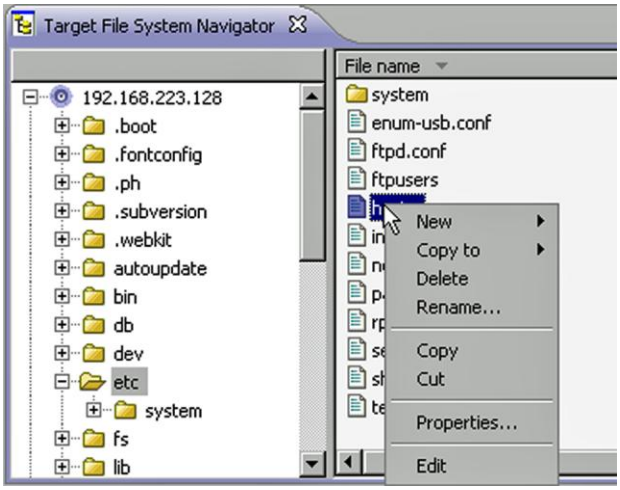

Figure 15: Working with files on the target system

Tip 16: Automated Header File Include

The Eclipse IDE supports automatic inclusion of header files. To know which `include` file your identifiers come from, select a function in your code and enter Ctrl+Shift+N, or Select Source > Add Include.

The IDE will edit your source file to insert the appropriate `include` file. For example, if you need an *`fopen()`* function with standard I/O and you do not already have one in your source file, the IDE will automatically insert one for you.

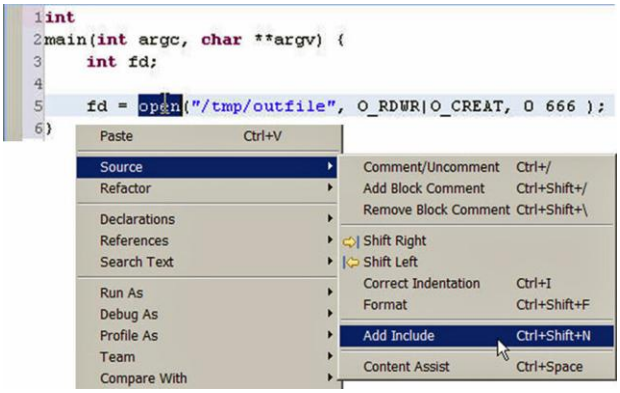

Figure 16: Configuring the IDE to automatically include header files

Tip 17: Block Editing

The Eclipse IDE supports block editing. To use block editing, all you have to do is select the block of code you want to change, and do one of the following, as needed:

- Use Tab or Backtab (Shift+Tab) to move the block left or right, as needed.
- Use Ctrl and the arrow keys to move the block up or down.

- Automatically comment out the whole block by entering Ctrl+/. This adds C++ comment delimiters "//" (slash-slash) around the selected block.
- To use C comment delimiters, enter Ctrl+Shift+/. This key sequence adds "/** ... **/", around the selected code.
- Reformat the code block to match the source coding style you select by entering Ctrl+Shift+F.

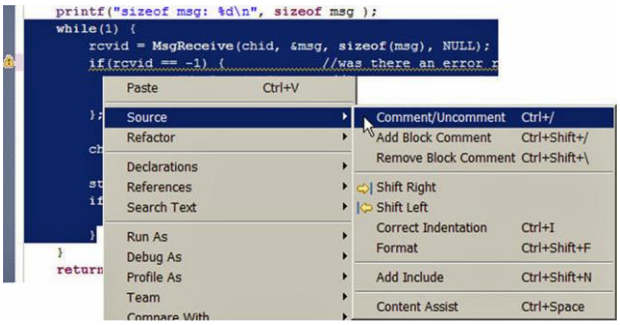

Figure 17: Using block editing

Tip 18: Reformatting Code

The Eclipse IDE includes configurable C/C++ code formatter with predefined styles. To use these styles:

1. Select Windows > Preferences > C/C++ > Code Formatter.
2. Choose one of K&R, BSD/Allman, GNU, Whitesmiths, or a custom style.

New code assumes the selected style. To apply a style to a code selection:

1. Select the code you want to format.
2. Enter Ctrl+Shift+F.

This feature offers flexible control of braces, whitespace, keywords, line wrap, and indentation.

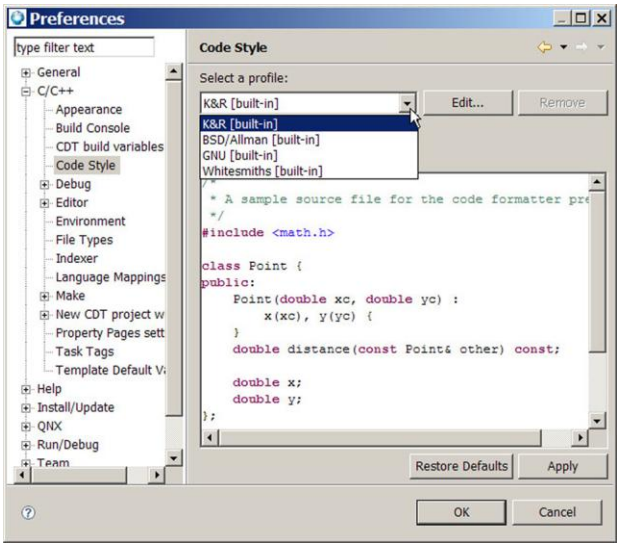

Figure 18: Reformatting code

Tip 19: Function Completion

One of the most popular time-saving features in the Eclipse IDE is Function Completion. To use this feature, simply enter the first characters of a function name, then Ctrl+Space to list matching functions.

As you enter more characters, the IDE narrows down the list of functions that match your entry. At any time, you can:

1. Select a function from the list.
2. Use the Enter key to have the IDE enter the selected function into your code.

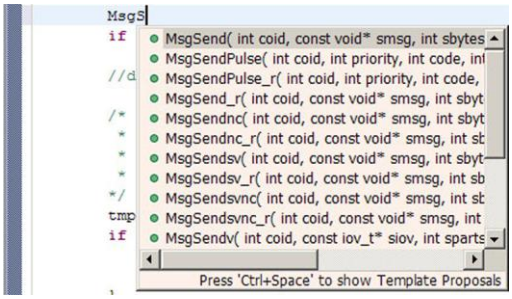

Figure 19a: The IDE displaying matching functions

After it has entered the selected function into your code, the IDE will prompt you for parameters, as shown in Figure 19b.

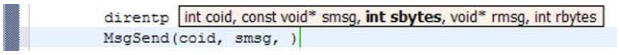

Figure 19b: The IDE prompting for function parameters

Tip 20: Automatic Structure Completion

The Eclipse IDE's Structure Completion feature is invoked just like the Function Completion feature, by typing the first characters of a structure name, then Ctrl+Space. It works in the same manner, offering a list of possible structures or unions to choose from, and providing element names and types.

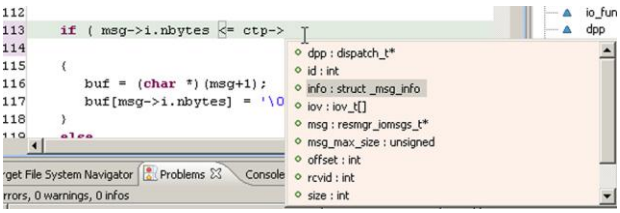

Figure 20a:

Automatic structure completion

You can configure the Eclipse IDE to automatically complete structures after a specified delay, as well as following specified key strokes.

To configure the delay:

3. Select Window > Preferences > C/C++ > Editor > Content Assist.
4. Enter the delay value, in milliseconds.

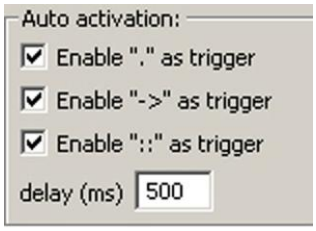

Figure 20b: Configuring automatic structure completion on delay

Tip 21: Prototypes, Definitions, and Implementations

The Eclipse IDE provides features that simplify working with functions:

- Highlighting a function and entering F3 will take you to the function.
- Hovering over a function and pressing F2 will open a read-only mini-editor with the function definition.

There is no need to browse to the file containing the function definition. See Tip 22: `#define` Expansion.

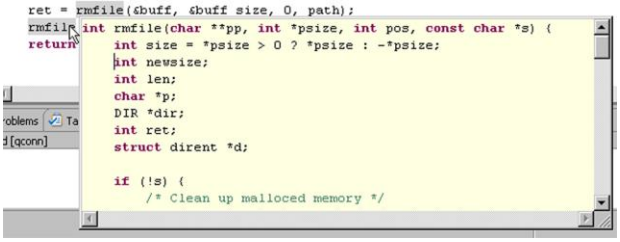

Figure 21: Viewing function information

Tip 22: `#define` Expansion

The `#define` expansion feature helps you understand what a `#define` actually evaluates to, and what the compiler inserts when it uses that `#define` statement.

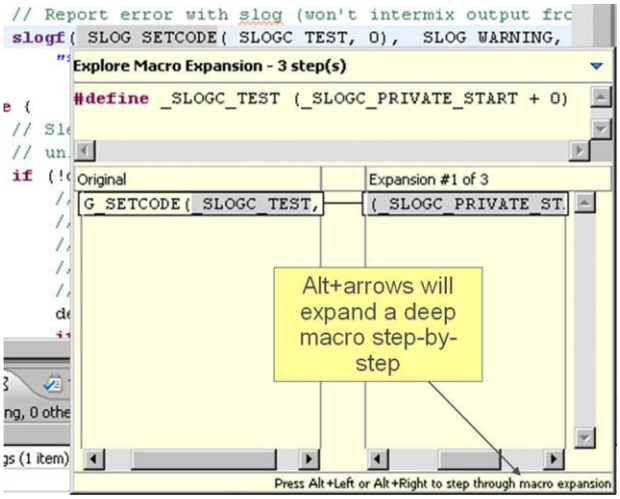

Figure 22: Expanding #defines

To use this feature:

1. Select an identifier set with `#define`.
2. Enter F2.

The IDE displays the definition, and allows you to browse step by step through the expansion. You can browse through every expansion of nested macros as you encounter them. This technique is also an excellent way to debug macros. It shows you what is actually in the code, so you troubleshoot the code rather than what you think is in the code.

Tip 23: Undo and Redo

The Eclipse IDE supports undo and redo editing. To undo your changes or to redo do them, before saving your file, enter Ctrl+Z to undo your last change, or Ctrl+Y to redo what you just undid.

By default, the IDE tracks the most recent 200 changes you have made to a file. You can change this value in your IDE Preferences.

Viewing original text

The Eclipse IDE places change bars in the margin of code you have changes since the last file save. To view the original code, that is, the code as it was at the time of the last file save, simply hover the mouse over the relevant change bar.

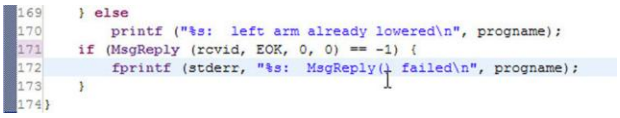

Figure 23a: Viewing change bars

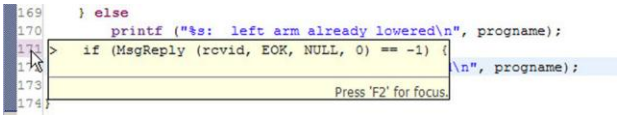

Figure 23b: Viewing original text

Tip 24: Local History

The Eclipse IDE keeps track of all changes made to a file since it was first saved. Local history is the record of these changes, which you can view to see all the specific changes that were applied to a file each time it was saved.

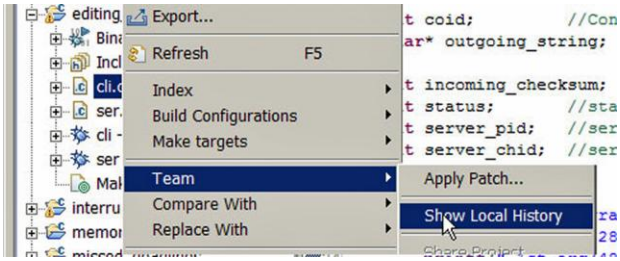

Figure 24a: Showing local history

To access local history:

1. Open the Project Navigator.
2. Select Team > Select Show Local History.

Comparing files

Though it is not necessarily associated with a code management system, local history is under the team menu, because it is akin to configuration management.

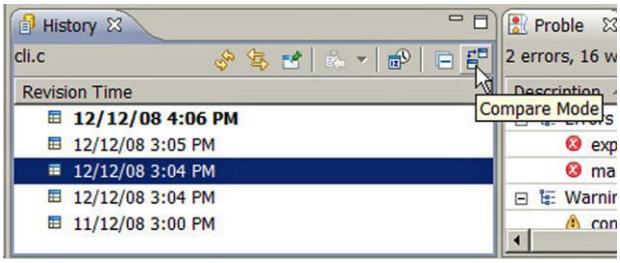

Figure 24b: Selecting compare mode

The local history feature is like having a mini-configuration management or a mini-source management tool, because even if you are not using source management, you can still compare any two saved versions of your file.

Further, if you have made multiple changes to a file, you can use a side-by-side view to compare files visually, then select the changes you want to keep and those you want to discard (Figures 13b and 13c).

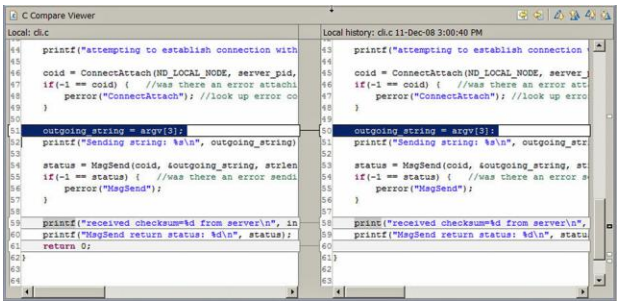

Figure 24c: Comparing file versions

Configuring local history

Saving local history can consume large amounts of disk space. However, you can configure the maximum amount of disk space the IDE will use for this feature:

1. Select Windows > Preferences.
2. Configure the disk space limits.

Tip 25: Quick Access

The Eclipse IDE's Quick Access pop-up offers an easy way to access items when you are not sure where they are. To use the Quick Access feature:

1. Enter Ctrl+3 to launch the pop-up.
2. Enter the term you want to search.

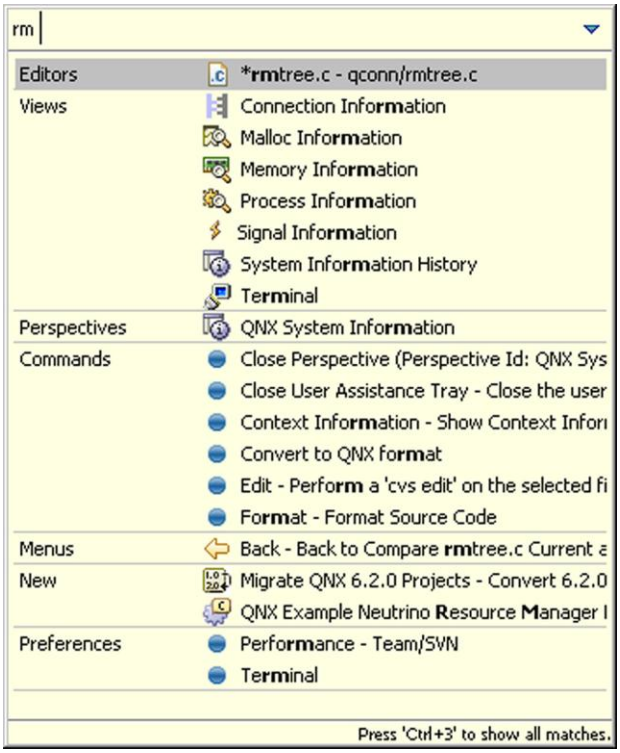

Figure 25: Using Quick Access

The IDE searches your entire workbench for editors, views, perspectives, menus, commands, or anything else that might contain the term you requested.

Tip 26: Code Folding

Code folding supported by the Eclipse IDE folds functions, structures and other entities into a single line to make it easier to read through code. To configure code folding:

1. Select Windows > Preferences > C/C++ > Editor > Folding.

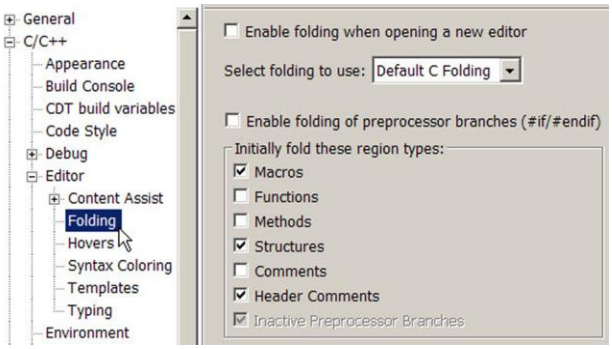

Figure 26a: Folding code

Working with code folding

The following commands speed working with code folding:

- Hover help reveals the contents of folder code
- Ctrl+/ ("/" on the number pad) toggles folding on/ off.
- Shift+Ctrl+/ ("/" on the number pad) folds all expanded code.

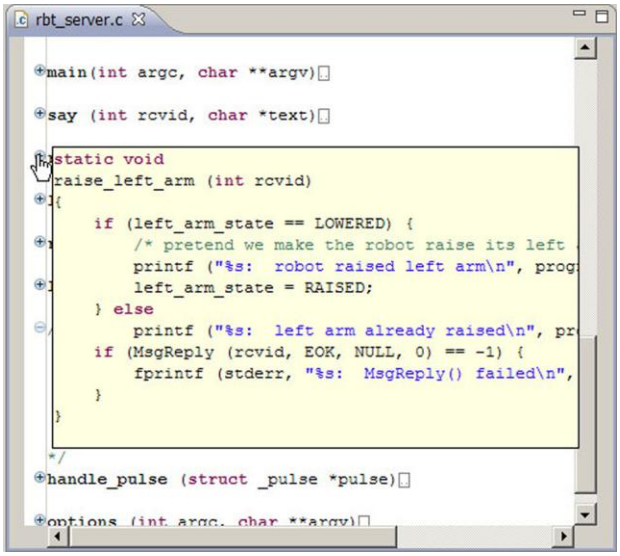

Figure 26b: Viewing the contents of folded code

Tip 27: Favorite Plug-ins

A particularity of Eclipse IDEs is that the basic product only contains a small set of core functionality; a large proportion of functionality is provided by plugins.

The following are plugins Eclipse IDE users have found particularly useful:

Mylyn — a task tracker with interfaces to Bugzilla & Trac (http://www.eclipseplugincentral.com/Web_Links-index-req-viewlink-cid-587.html)

Grep Console — provides regular expression matching on console output (http://www.eclipseplugincentral.com/Web_Links-index-req-viewlink-cid-1275.html)

SVN — plugins for Subversive (http://www.eclipse.org/subversive), and Subclipse (http://subclipse.tigris.org)

NTail — Dynamic log file tail (http://www.certiv.net/downloads/ntaildownload.html)

RSS View — RSS reader for bug tracking, developer forum, wikis, etc. (http://www.eclipseplugincentral.com/Web_Links-index-req-viewlink-cid-369.html)

Getting the Eclipse IDE

If you do not already have the Eclipse IDE and want to try it, you can download an evaluation copy of the Eclipse Momentics Tool Suite, which offers a full embedded C/C++ development environment, from www.qnx.com/products/evaluation/ .

You can also obtain an Eclipse IDE from Eclipse.org directly at www.eclipse.org. If you choose this option, you will have to look after a few things yourself. You will need to:

1. Supply your GCC tool chain, with your compiler, your linker, the GDB debugger, and all of the other components you may need.
2. Install the CDT plug-in, for C or C++ development tools.

Once you have the plug-in and the tools and you have them configured, you will be able to start using your Eclipse IDE for your embedded system development work.