": QNX

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

Introduction1
About Eclipse1
Contents
Tip 1: Show Key Assist
Tip 2: Key Binding
Tip 3: Nice-to-know Keyboard Shortcuts4
Tip 4: Refactoring4
Tip 5: Call Hierarchy5
Tip 6: Hyperlink Navigation6
Tip 7: Bookmarks6
Tip 8: Prompt for Arguments During Launch7
Tip 9: Template Proposals8
Tip 10: View Assembly Code9
Tip 11: Detaching Views10
Tip 12: Group Launch10
Tip 13: Directory Path Variables11
Tip 14: Custom Breakpoint Actions12
Tip 15: Manipulating Target Files13
Tip 16: Automated Header File Include14
Tip 17: Block Editing14
Tip 18: Reformatting Code15
Tip 19: Function Completion16
Tip 20: Automatic Structure Completion16
Tip 21: Prototypes, Definitions, and Implementations
Tip 22: #define Expansion17
Tip 23: Undo and Redo18
Tip 24: Local History19
Tip 25: Quick Access
Tip 26: Code Folding20
Tip 27: Favorite Plug-ins21
Getting the Eclipse IDE

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.

Add Include	Ctrl+Shirt+N
Backward History	Alt+Left
Build All	Ctrl+B
Close	Ctrl+W
Close All	Ctrl+Shift+W
Collapse	Ctrl+Numpad_Subtract
Collapse All	Ctrl+Shift+Numpad_Divide
Comment/Uncomment	Ctrl+/
Commit	Ctrl+Alt+C
Content Assist	Ctrl+Space
Context Information	Ctrl+Shift+Space
Сору	Ctrl+C
Copy Lines	Ctrl+Alt+Down
Create Patch	Ctrl+Alt+P
Cut	Ctrl+X
Debug	F11
Delete	Delete
Delete Line	Ctrl+D
Delete Next Word	Ctrl+Delete
Delete Previous Word	Ctrl+Backspace
Delete to End of Line	Ctrl+Shift+Delete
Duplicate Lines	Ctrl+Alt+Up
Expand	Ctrl+Numpad_Add
Expand All	Ctrl+Numpad_Multiply
Explore Macro Expansion	Ctrl+=
Extract Constant - Refactoring	Alt+C
Extract Function - Refactoring	Alt+Shift+M
Find Declaration	Ctrl+G
Find Next	Ctrl+K
Find Previous	Ctrl+Shift+K
Find References	Ctrl+Shift+G
Find Text in Workspace	Ctrl+Alt+G
Find and Danlara	
Press "C	tri+Shift+L" to open the preference page

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.

ieys 🔤	Keys				• • •
General	Scheme: Default				
	type filter text				
	Command ~	Binding	When	Category	User
	Build All	Ctrl+B	In Windows	Project	
	Build Automatically			Project	
	Build Clean Build Brotect	Childe	In Windows	Project	01
	C/C++ Content Assist (type:	Basic Propo	111111111111	Edit	
	C/C++ Content Assist (type:	Help Propo:			
	C/C++ Content Assist (type:	Parsing-bas		Conflicting key	/
	Les e l'united	d and a second		sequences	
Click in bindin	Cl Unoind Command	Restore Command		shown here	
Click in bindin	9 mar Build Project		_	onown nore	
window and	scription: Build the selected project	t.	Con(licts:	1	
press key			Command	When	
sequence			Build Project	In Window	5
	Binding: Ctrl+Pl	19	Print	In Window	5
	Internet Training		5		
	man primions	10 10 10 10 10 10 10 10 10 10 10 10 10 1			
				Fiters	Evport
	Cant	and an loss		- Possini	- Coportin
	Conte	ext when		Restore Defaults	Apply
	shortcu	t operates			

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.

) g->off	Rename global function 'line_bind'		Select files to	- 0 ×
	; if((g->wil	Changes to be performed		apply refactor	*
) int	g->wil write() line bind(struct gat	Rename global function line, bind D Configuration of the second geometry of t	Ste previ	ep through change ew before accepting	
	void	Children & Source	Def	- 40 10	TO NA
	<pre>static con char oldfunc = if(fd_info g = ol free(g</pre>	<pre>crigina double "iffine_bind(id, apointo_cmd_handler, 0, ah->global- return LIME_ERROR(errno); iff(perms) { apointo_cmd_handler(id, parms, flags, 0, 0, 0); return LIME_NEXT; }</pre>	× if	Rubel Joack gosinto cmd handler, 0, ah->g return LINE_ERROR(errno); (parms) (cmd_handler(fd, parms, flags, 0, 0, eturn LINE_NEXT;	0.
) else (*Name: hanlmintry attach	* N	ame hanimiter stach	•

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;
                      err = 0;
    int
    while((c = getopt(argc, argv, "p:d:")) != -1) (
    switch(c) {
         case 'd':
             if(registry_set("debug_device", optarg, 0) == -1) (
                 return -1;
             3
             break;
         default:
             err = 1;
             break;
        3
    3
```

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.

Bookmar show in ma	ks argin ar *argv[]) (
int c; int err = 0 Togde Breakpoint Disable Breakpoint: Breakpoint Properties	argv, p.	Right-click nargin to w bookm	cin add arks
Add Boolymark	"debug_de	vice", opt	arg, 0) ==
Show Quick Diff Ctrl+Shift+Q Show Line Numbers Folding		Bookr wind	nark Iow
Preferences			
4 items	*		
Description A	Resource	Path	Location
bind entry point	fds.c	qconn	line 69
main	main.c	qconn	line 27
parse_args	args.c	qconn	line 21
registry_set malloc code	registry.c	qconn	line 55

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.

ame: pea	k config (query)			
O Main	🕽 = Arguments 🔪 🊾 Enviro	nment 👩 Upload) 🎋 Debugger	Source
C/C++ Pro	gram Arguments:			
\${string_	rompt}			
1000 B	ashawi an kaynahi			

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.

🗿 Yariable input		×
Please input a value		
0x30003		
	ОК	Cancel

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.

Jame	Context	Description	Auto Inc		New
author	Comment	author name	Adio 115		1101111
addrior acatch	CIC++	catch block	00		Edit
	CIC++	class declaration	00		
	CIC++	default multiline comment	00		Remove
do do	CIC++	do while statement	00		
7 else	C/C++	else block	on	Rec	
elseif	C/C++	else if block	on		eres in a manual series of
for	C/C++	for loop	on	Res	vert to Defa
of for	C/C++	for loop with temporar	on		
2 if	C/C++	if statement	on		*
ifelse	C/C++	if else statement	on		Import
z main	C/C++	main method	on		Export
Inamespace	C/C++	namespace declaration	on		Export
new 2	C/C++	create new object	on		
stderr	C/C++	print to standard error	on		
stdout	C/C++	print to standard output	on	-	
evie <u>w</u> : or (\$/ver) = 0 · \$	(ver) < \$(mev) + ti	S(ver))		
\${line	selectio	on) \$ (cursor)	(((al)))		
- 1	aword Ctrl+S	ace Tab wal	ks through a	all fields	

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.

Project Ex 🛛 🗖 🗖	agent.		6	line.c	14	C	men	nspeed.c	C an	gs.c	peek.c	peek.	2 23
= 😫 🏹	43a:	e8	fc	11	11	11			call	43b	<pre><pre>cpeek+0x97</pre></pre>	7>	
⊕ 192.168.223.128			4	3b:	R	386	PC	32 ex	it				
🗄 🚰 dir2xfs	43f:	83	c4	10	177		2		add	\$0x1	0,tesp		
🗈 🥵 filewatch	442:	89	16						mov	<esi< td=""><td>,%esi</td><td></td><td></td></esi<>	,%esi		
E ElashToRam	444:	c9							leave				
🗄 😂 imagemanip	445:	c3							ret				
E Media5200	446:	89	16						mov	4esi	,%esi		
E C memspeed													
E C peek	0000044	8 <	par	sea	rg>	:							
🖻 🐝 Binaries	448:	55							push	*ebp			
1 Includes	449:	89	e5						mov	tesp	,tebp		
🕀 🕞 arm	44b:	83	ec	18					sub	\$0x1	8,%esp		
E Co mips	44e:	8b	45	0c					mov	Oxc (tebp),teax	4	
E Ca ppc	451:	8b	10						mov	(tea	x),%edx		
1 Co sh	453:	8d	04	95	00	00	00	00	lea	0x0 (, tedx, 4), 1	eax	
E-C- x86	45a:	8b	55	08					mov	0x8 (tebp),ted	4	
B-CB-0	45d:	8b	04	02					mov	(ted	x, teax, 1),	*eax	
E O neek - [x86	460:	40							inc	*eax			
F- an neek.o - [x	461:	8a	10						mov	(tea	x),%dl		
Makefie	463:	80	c2	cf					add	\$0xc	f, adl		
B-C 0-0	466:	0f	be	c2					movsbl	\$d1,	*eax		
E to neek a - fx	469:	83	1 8	30					cmp	\$0x3	c, teax		
F-m peek o - [y	46c:	Of	87	6e	01	00	00		ja	5e0	<parsearg+< td=""><td>+0x198></td><td></td></parsearg+<>	+0x198>	
Makefie	472:	8b	04	85	e4	00	00	00	mov	0xe4	(, teax, 4),	*eax	
Makefile			4	75:	R	386	32	.r	odata				
F-C peek.c	479:	ff	eO						jmp	#tea	x		
- Common.mk	47b:	90							nop				
Makefile	47c:	c7	05	00	00	00	00	01	movl	\$0x1	,0x0		

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:

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.

🕽 New Folder		_ 🗆 >
Folder	e e e e e e e e e e e e e e e e e e e	
Enter or select the parent fol	der:	
tst_server		
☆ ← ⇔		
🖭 😂 tfs-cp		
E 😂 tst_server		_
± ≥ x86		•
Folder and a large t		
rolder mame: Toucpuc		
<< <u>A</u> dvanced		
Link to folder in the file s	vstem	
SOURCE DIR output	Browse	Variables
Developed learning Colored		
Reserved location. Crippi	. pour costou put	
0	Finish	Cancel
U	Cullsti	

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.

Name Print loop status Untitled Sound Action	Type GDB Command Action Sound Action	Summary print status C:\WINDOWS\Medi
emove		Up Dow
Vame	Туре	Summary
Print loop status Untitled Sound Action	GDB Command Action Sound Action	print status C:\WINDOWS\Medi

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.

🔁 Target File System Navigator 🛛	
192.168.223.128 .boot .fontconfig .ph .subversion	File name system enum-usb.conf ftpd.conf ftpusers New
	☐ in Copy to → ☐ n' Delete p Rename
⊕… ் dev ⊡- '	I s Copy I st Cut I te Properties
🗼 🕀 🛅 lib	Edit

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.

while(1) rcv if() { id = MsgReceive(chid, &ms rcvid == -1) {	1, s: //w	zeof(msg), NULL); as there an error r	
	Paste Ctrl+V			
};	Source	Þ	Comment/Uncomment	Ctrl+/
ch	Refactor	•	Add Block Comment	Ctrl+Shift+/
	Declarations		Remove Block Comment	Ctrl+Shift+\
st	References	•	Shift Right	
if	Search Text	•	😂 Shift Left	
	Run Ac		Correct Indentation	Ctrl+I
3	Debug As		Format	Ctrl+Shift+F
return	Profile As		Add Include	Ctrl+Shift+N
_	Team	•	Content Assist	Ctrl+Space

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.
- Use the Enter key to have the IDE enter the selected function into your code.

if	MsaSend(int coid, const void* smsa, int shutes)
	 MsgSend(int cold, collsc void sinsy, int solves - MsgSendBulse(int sold, int priority, int solves -
110	 MsgSendPulse (Int cold, Int priority, Int code, Int
114	 MsgSendPulse_r(int cold, int priority, int code,
100	MsgSend_r(int coid, const void* smsg, int sbyt
/*	 MsgSendnc(int coid, const void* smsg, int sbyt
*	MsgSendnc_r(int coid, const void* smsg, int st
*	 MsqSendsv(int coid, const void* smsq, int sbyt—
*	MsgSendsv_r(int coid, const void* smsg, int sb
/	 MsgSendsvnc(int coid, const void smsg, int st
tmp	MsgSendsvnc_r(int coid, const void* smsg, int
if	MsgSendv(int coid, const iov_t* siov, int sparts -
	<
	Press 'Ctrl+Space' to show Template Proposals

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:

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

Auto activation:
🔽 Enable "." as trigger
🔽 Enable "->" as trigger
🔽 Enable "::" as trigger
delay (ms) 500

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

progname);
", progname);

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.

	🖻 🤔 editing	🛃 Export	3	t coid; //Con
I	🕀 🐝 Bina	🐑 Refresh 🛛 🕞 F5		ar* outgoing_string;
	 ⊕ C Cli. <li< td=""><td>Index Build Configurations Make targets</td><td>• •</td><td><pre>t incoming_checksum; t status; //sta t server_pid; //ser t server_chid; //ser</pre></td></li<>	Index Build Configurations Make targets	• •	<pre>t incoming_checksum; t status; //sta t server_pid; //ser t server_chid; //ser</pre>
I	Mal	Team	►	Apply Patch
		Compare With Replace With	+	Show Local History 28
	the missed		_	Charo Project

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.

His	tory 🛛 🖓 🖓	Proble 🕅
cli.c		2 errors, 16 w
Revisi	on Time	Description
	12/12/08 4:06 PM	mpare Models
	12/12/08 3:05 PM	🔕 exp
	12/12/08 3:04 PM	😣 ma
	12/12/08 3:04 PM	🖃 🗄 Warnii
	11/12/08 3:00 PM	, 🇥 cor
		•

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).

C Compare Viewer	÷ 🗟 🗧 🔬 🖓 🖓			
Local: di.c	Local history: cli.c 11-Dec-08 3:00:40 PM			
<pre>temp:intf("attempting to establish connection with de coold = ConnectAttach(ND_LOCAL_NODE, server_pid, ff(-1 == coid) { //was there an error stachl perror("ConnectAttach"); //look up error co fs sc</pre>	<pre>c3 printf("attempting to establish connection (A) coid = ConnectAttach(ND_LOCAL_NODE, server_) c6 if(-1 == ooid) { //was there an error att c7 perror("ConnectAttach"); //look up erro c8 }</pre>			
 cutcoincg string = x try [3]; printf("Sending string: % hin", outgoing_string) status = MagSend(soid, sourgoing_string, string) status = MagSend(soid, sourgoing_string, string) perror("MagJend"); perror("MagJend"); 	<pre>0</pre>			
59 print("received checksum*&d from server'n", in- co print("Hegdend return status: %d\n", status); 61 return 0; 63 64	S print("received checksum*id from server\u", printf("NsgSend return status: 6d\n", statu 60 61 62 63 64 55 •			
x	•			

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.

Editors	c *rmtree.c - qconn/rmtree.c
Views	Connection Information
	🐼 Malloc Information
	👼 Memory Info rm ation
	🖏 Process Info rm ation
	Signal Information
	🐻 System Info rm ation History
	Jerminal
Perspectives	🐻 QNX System Information
Commands	Close Perspective (Perspective Id: QNX System)
	Close User Assistance Tray - Close the use
	Context Information - Show Context Information
	Convert to QNX format
	Edit - Perform a 'cvs edit' on the selected i
	Format - Format Source Code
Menus	😓 Back - Back to Compare rmtree.c Current
New	Migrate QNX 6.2.0 Projects - Convert 6.2.0
	🧐 QNX Example Neutrino Resource Manager
Preferences	Performance - Team/SVN
	🔵 Terminal

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)

QNX Software Systems

- **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.

About QNX Software Systems

QNX Software Systems Limited, a subsidiary of BlackBerry, is a leading vendor of operating systems, development tools, and professional services for connected embedded systems. Global leaders such as Audi, Cisco, General Electric, Lockheed Martin, and Siemens depend on QNX technology for vehicle infotainment units, network routers, medical devices, industrial automation systems, security and defense systems, and other mission- or life-critical applications. Founded in 1980, QNX Software Systems Limited is headquartered in Ottawa, Canada; its products are distributed in more than 100 countries worldwide. Visit <u>www.qnx.com</u> and <u>facebook.com/QNXSoftwareSystems</u>, and follow <u>@QNX New</u>s on Twitter. For more information on the company's automotive work, visit <u>gnxauto.blogspot.com</u> and follow <u>@QNX Auto</u>.

www.qnx.com

© 2013 QNX Software Systems Limited. QNX, QNX CAR, Momentics, Neutrino, Aviage are trademarks of QNX Software Systems Limited, which are registered trademarks and/or used in certain jurisdictions. All other trademarks belong to their respective owners.