

Quincy



for Pawn

May 2011

“CompuPhase” is a registered trademark of ITB CompuPhase.

“Microsoft” and “Microsoft Windows” are registered trademarks of Microsoft Corporation.

“Linux” is a registered trademark of Linus Torvalds.

“Quincy” was the name of the cat of Al Stevens’ daughter; Al Stevens is the author of the original versions of quincy.

Copyright © 2002, Al Stevens.

Portions copyright © 2008–2011, ITB CompuPhase, [www.compuphase.com](http://www.compuphase.com)

The information in this manual and the associated software are provided “as is”. There are no guarantees, explicit or implied, that the software and the manual are accurate.

Requests for corrections and additions to the manual and the software can be directed to ITB CompuPhase at the above address.

Typeset with T<sub>E</sub>X in the “Computer Modern” and “Palatino” typefaces at a base size of 11 points.

---

## Contents

INTRODUCTION .....	1
What Does the Name Quincy Mean? .....	1
STARTING AND STOPPING QUINCY .....	2
Exiting Quincy .....	2
SETTING OPTIONS .....	4
Build Options .....	4
Debugger / Logging Options .....	6
Editor Options .....	8
Snippet Options .....	9
Miscellaneous Options .....	9
Saving your option settings .....	11
CREATING A PROGRAM .....	12
Creating a source code file .....	12
Saving a source code file .....	13
Using Save As .....	14
Working with multiple source code & include files .....	14
EDITOR INTERFACE .....	16
Entering Text .....	16
The Insertion Cursor .....	17
Brace matching .....	17
Breakpoints as bookmarks .....	18
The toolbar .....	18
Info-tips .....	19
Code snippets .....	19
Text completion .....	20
Searching and replacing text .....	21
The symbol browser .....	21
KEYBOARD SHORTCUTS .....	23
Editor keys .....	23
Debugger keys .....	24
Miscellaneous keys .....	25
TARGET HOST CONFIGURATION FILE .....	26
INDEX .....	27



## Introduction

---

---

“Quincy” is an integrated development environment (IDE) that integrates a programmer’s editor, compiler, and debugger into one Microsoft Windows application. This version of Quincy is adapted to the PAWN scripting language.

A stripped-down version of Quincy is available for Linux; it is called wxQuincy. This guide applies to both, but many of the features and options of Quincy are not available in wxQuincy.

Al Stevens wrote Quincy as a teaching tool distributed on CD-ROM with books that he wrote about C and C++. Al Stevens has since retired from the programming and book authoring business, but his work lives on. The original versions of the Quincy IDE are still available.

This document assumes that you already know how to run Windows and its applications. Much of the information presented here has its roots in the manual that Al Stevens wrote for Quincy.

### What Does the Name Quincy Mean?

Quincy is named after that cat of Al Stevens’ daughter Wendy. As a child in the 1970s, Wendy was a fan of the TV shows, *The Odd Couple* and *Quincy*, both of which starred Jack Klugman, who played Oscar Madison and a medical examiner named Quincy in those shows. When Wendy brought home a tiny white kitten with blue eyes, she didn’t know its sex. Choosing from Jack, Oscar, and Quincy, she decided that Quincy was the most gender-unspecific.

Quincy lived many years and was a beloved pet to Wendy as a child, Mr. and Mrs. Stevens when Wendy went to college, and then to Wendy and her family after Wendy married and had children of her own.

Oh, yeah, Quincy was a female.

## Starting and stopping Quincy

---

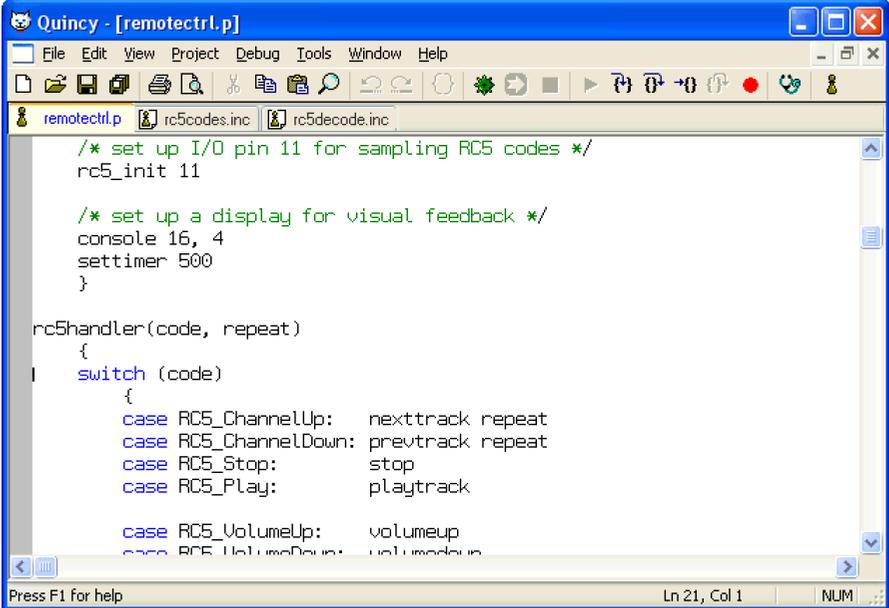
There are two ways to start Quincy:

1. Open the Quincy submenu on the Start/Programs menu.
2. Click the Quincy command.

Or:

1. Double-click the Quincy icon that the Setup program optionally added to your desktop.

Quincy always loads the source code files from your most recent Quincy session. The following screen shot shows the Quincy application window with a source code file loaded and ready to go. (The very first time you start Quincy, there will be no source code document loaded.)



The screenshot shows the Quincy application window titled "Quincy - [remotectl.p]". The window has a menu bar with "File", "Edit", "View", "Project", "Debug", "Tools", "Window", and "Help". Below the menu bar is a toolbar with various icons. The main area displays a source code file with the following content:

```

remotectl.p rc5codes.inc rc5decode.inc

/* set up I/O pin 11 for sampling RC5 codes */
rc5_init 11

/* set up a display for visual feedback */
console 16, 4
settimer 500
}

rc5handler(code, repeat)
{
  switch (code)
  {
    case RC5_ChannelUp:  nexttrack repeat
    case RC5_ChannelDown: prevtrack repeat
    case RC5_Stop:       stop
    case RC5_Play:       playtrack

    case RC5_VolumeUp:   volumeup
    case RC5_VolumeDown: volumedown
  }
}

```

The status bar at the bottom of the window displays "Press F1 for help", "Ln 21, Col 1", and "NUM".

## Exiting Quincy

To exit from Quincy, use one of the following:

1. choose the Exit command on the File menu,
2. or click the × button in the upper right corner of the application title bar,
3. or press **Alt+F4**

If any documents have been changed and not saved, a dialog box asks if you want to save them before exiting. The dialog box asks the question once for each modified document.

## Setting Options

---

---

There are several options for how a PAWN program is to be “built” from source code into an executable form. You set these options prior to compiling and running the program. Quincy remembers the option settings from session to session.

If you are using Quincy with the tutorial examples as described in the section **Example Programs**, the tutorial “project files” set the options that each example uses. You can override these options during the session, but the tutorial command files are not affected when you do so. The next time you open the tutorial, its programmed options are restored.

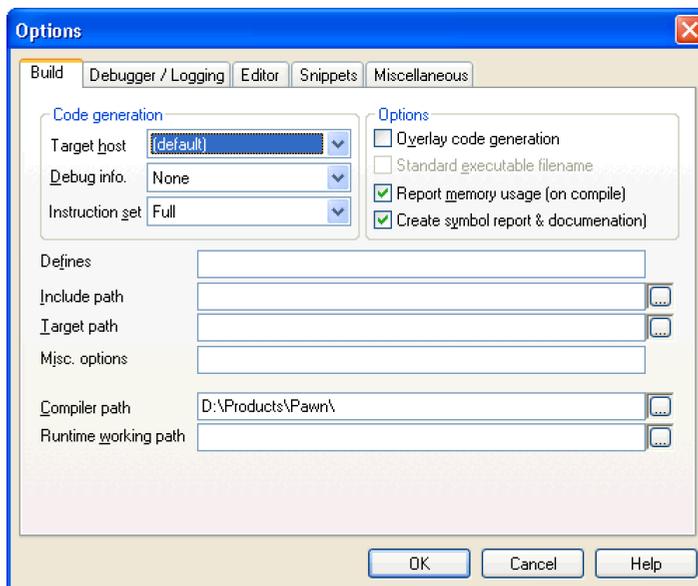
To set Quincy’s options choose the **Tools** menu option and then the **Options...** sub-menu. Quincy opens the options dialog, a tabbed dialog with tabs for **Build** options, **Run** options, **Editor** options and **Miscellaneous** options.

### Build Options

These are the options that control how Quincy builds a program, including how the PAWN compiler processes the source code, what kind of code is generated, and where the compiler looks for header files and libraries.

To review and change the Build options, select the **Build** tab of the **Options** dialog as shown above.

- ◇ When building for a specific target, select it in the **Target host** field. This setting reconfigures balloon help, list of examples and the configuration of the PAWN compiler. See the section **Target host configuration file** on [page 26](#) for details.
- ◇ Select the level of **Debug info.** for the program, depending on whether you want to use the IDE to debug the program. For a final build, especially for a platform with memory constraints, set the debug information to **None**.
- ◇ Select the **Instruction set** that is supported by the host(s) that the script must run on. The PAWN abstract machine has a ‘core’ instruction set that is supplemented by two additional sets. A host may implement the full instruction set, or only a smaller set. The Just-In-Time compiler (JIT), for example, supports only the “Core” instruction set. When the compiler can generate code for a larger instruction set, it creates more compact and quicker code.

FIGURE 1: *Build options dialog*

- ◇ When writing large scripts that must fit in little memory, check the option **Overlay code generation**. Overlay code is *incompatible* with the Just-In-Time compiler (JIT).
- ◇ If a host application or environment requires this, you may need to set the **Standard executable filename** option.
- ◇ For a summary on how much memory the script uses, set the option **Report memory usage**.
- ◇ The option **Create symbol report & documentation** creates a detailed report for both the symbol browser and documentation generation. The report is augmented with the contents of the “documentation comments” in the source code of the script. This report is in an XML file, which can be viewed in a web browser.
- ◇ Enter any preprocessor macros that the program needs in the **Define** field. Whatever you type here will be treated as if you put it at the front of the source code as the arguments to a `#define` preprocessor directive with this exception: to define a global symbol with a value, use an equals sign (=) as shown here:

```
ScriptVersion=2.5
```

The example just shown compiles as if the source code file included this statement:

```
#define ScriptVersion 2.5
```

If you need more than one macro, separate them in the “Define” field with a space character as shown here:

```
ScriptVersion=2.5 QueueSize=50
```

- ◇ Enter paths to folders where the PAWN compiler can search for header files specified with the `#include <file.inc>` directive. Separate the paths with semicolons. You can use relative paths or fully qualified paths. The compiler searches the paths in the order you enter them here. If the header file is not in one of these paths, then it searches the compiler’s standard directories for include files.
- ◇ Set the path where the target file has to be generated, if you wish this path to be different from the location of the source file(s).
- ◇ Enter any additional command line arguments that Quincy should pass to the PAWN compiler program. For example, adding `-c1252` causes Quincy to interpret the source code in the codepage 1252 (Latin-1).
- ◇ Enter the path where the PAWN compiler is installed. If you leave this field blank, Quincy looks for the compiler in the “/bin” subdirectory under the directory where the PAWN toolkit is installed.

For the Include path, Target path and Compiler path fields you can click the browse button (“...”) at the right of these fields to open the Browse for Folder dialog shown in figure 2. Browse to the folder you want to add to the list in the option dialog’s Build tab, and click OK.

## Debugger / Logging Options

Select the Debugger / Logging tab on the Options dialog as shown here.

When you wish to use the debugger, also make sure that you build the script with “debugging information”, see the section **Build options**.

- ◇ In most cases, the script that you wish to debug runs on the same system as the one that you are compiling on. This is known as “local debugging”. Quincy also supports “remote debugging”, where the script runs on one system and Quincy’s debugger on another system.

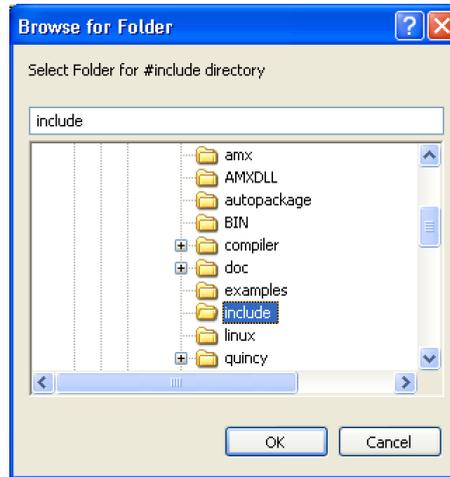


FIGURE 2: *Folder browse dialog*

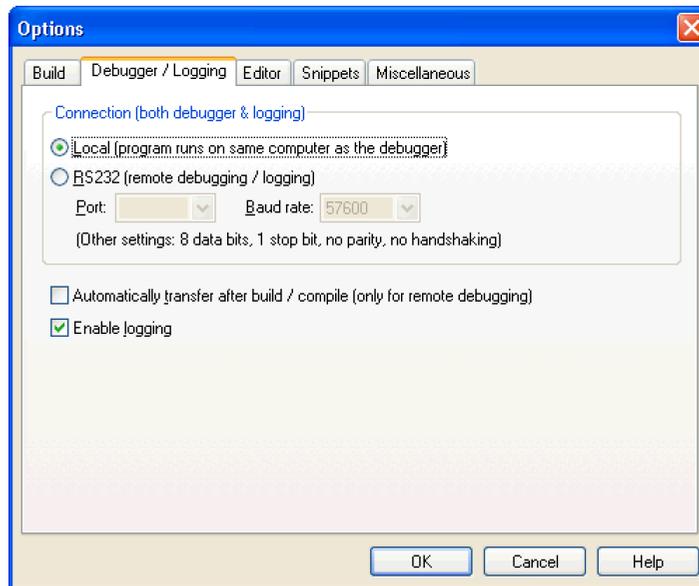
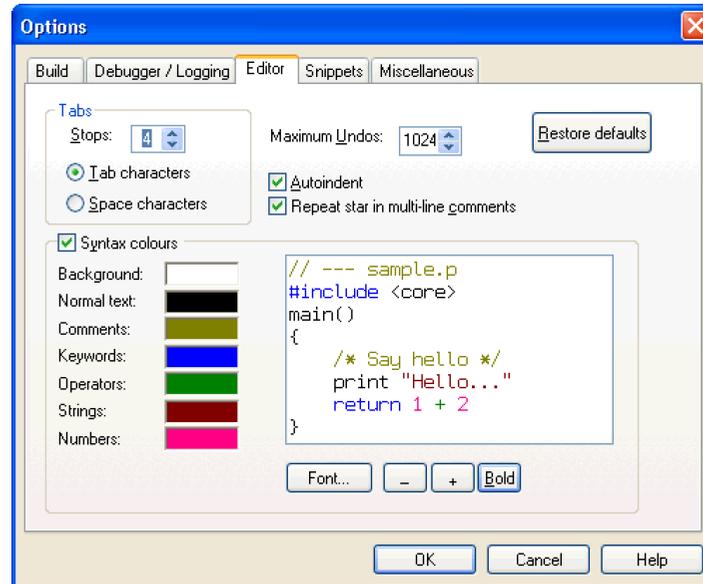


FIGURE 3: *Debugger options dialog*

- ◇ The two systems must be connected with a serial cable. You must also select which port to use and what baud rate the serial transfer uses.

FIGURE 4: *Editor options dialog*

- ◇ When using remote debugging, Quincy can optionally use the same serial line to transfer the script to the remote system. The option **Automatically transfer after build/compile** lets Quincy transfer the script immediately after a successful compile. If this option is unchecked, you can still transfer the script over the serial line explicitly.
- ◇ The serial connection is also used for logging information that a script may send over the serial line. This is particularly useful if a script runs on a remote device and that remote device does not support a debugger. Activate serial logging by checking the option **Enable logging**.

## Editor Options

Select the **Editor** tab on the **Options** dialog as shown here.

As you change the editor's settings, the simulated source code display in the **Editor** tabbed dialog reflects your changes.

- ◇ Select the number of character positions for each tab stop by changing the scroll button control.

- ◇ Select whether the editor inserts space characters or tab characters into the text when you press the **Tab** key or use the autoindent feature.
- ◇ Select the maximum number of edit action undos that Quincy stores.
- ◇ Enable or disable the autoindent feature by clicking the **Autoindent** check box control. With autoindent disabled, Quincy always returns the insertion cursor to the left margin when you press **Enter**.
- ◇ When creating `/* ... */` style comments over multiple lines, Quincy repeats the leading “`*`” on every new line if the option **Repeat star in multi-line comments** is checked.
- ◇ Enable or disable syntax color highlighting.
- ◇ With syntax color highlighting:
  - Click one of the color bars next to the text feature you want to highlight with color. This action opens the **Color** dialog shown here.
  - Click the color you want to use for the text feature selected.
  - Click **OK**.
  - To revert to Quincy’s default color scheme, click **Restore Defaults**.
- ◇ Change the font style with the **Font...** button, the font size size by clicking the **+** and the **-** buttons, and click the **Bold** button to toggle the font between normal width and boldface.

## Snippet Options

See the section on code snippets, on [page 19](#), for details on how to configure snippets and how to use them in the editor.

## Miscellaneous Options

Select the **Miscellaneous** tab on the **Options** dialog as shown here.

- ◇ Quincy can verify whether updates are available. If the option **Check whether updates are available online** is set, Quincy will access the Internet approximately once a week to look for updates. You can also ask Quincy to verify immediately, with the button **Check now**.

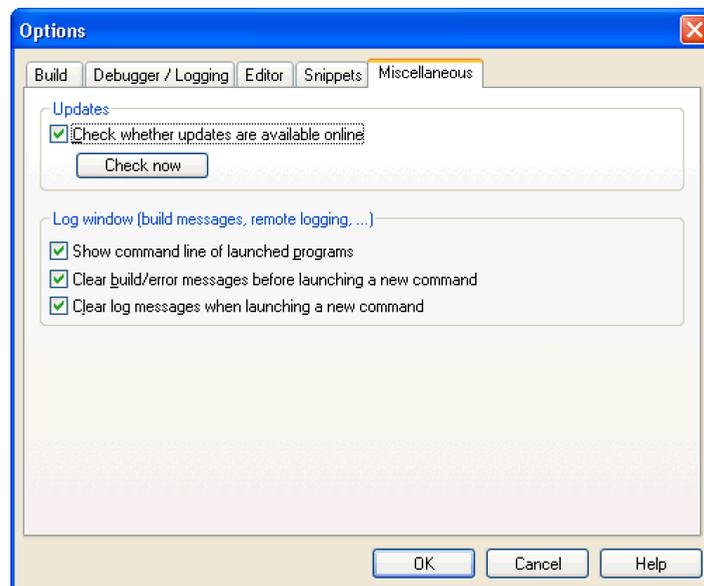


FIGURE 5: *Miscellaneous options dialog*

- ◇ In the “Build result” view that appears when compiling a program or transferring a program, the name of the program (and its options) that Quincy launches to do the actual work may optionally be included. To display this information, set the option **Show command line of launched programs**. To suppress this information, clear this option.
- ◇ The “Build result” view may keep the results of multiple compilation/transfer sessions in an ever growing log, or it may show only the results of the most recent build action, depending on the state of the option **Clear build/error log before launching a new command**.
- ◇ Similarly, the “Logging” view may keep the results of all data that has been received, or it may show only the data that is received since the most recent file transfer, depending on the state of the option **Clear log messages when launching a new command**. This is related to the serial logging option in the **Debugger / Logging** tab.

## Saving your option settings

- ◇ When you have made all the options changes you want to make, click OK on the Options dialog.
- ◇ Or, to nullify any changes you might have made, Click Cancel on the Options dialog.

If you are working with “workspace files” any options that you changed are stored in the workspace as well.

## Creating a program

---

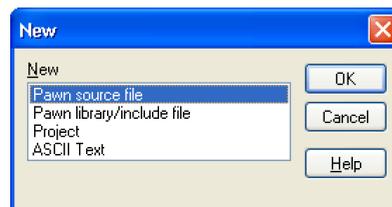
---

You write a new program by building a source code file, setting options for the program, saving the program to disk, and then compiling the program. First you must create a new file.

### Creating a source code file

To create a source code file from scratch, choose the **New** command on the File menu (alternatively, click the “New” tool button or type **Ctrl+N**).

Quincy opens the New file. . . dialog shown here.



Then select the kind of source code file you are building. A PAWN source file will be saved with the `.p` file extension and will be compiled by the PAWN compiler. An include file (or source code module) will be saved with the `.inc` extension, signifying a header file you include in your PAWN source code files.

After you click **OK**, Quincy creates an empty text file with the name `Textn`, where `n` is the next available number for Quincy to assign to a text file. You will probably want to change the name to something more meaningful when you save the file. This file has not been saved to disk yet. You cannot compile the program until you have saved the source code file.

Now you can use the text editing commands discussed in section **Editing source code** to write your program.

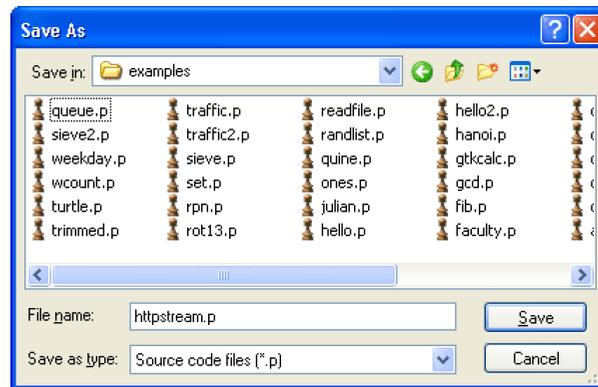
If you scroll the list in the New dialog, you will see that Quincy supports editing one more kind of files: ASCII files. ASCII text files are plain text files. Quincy will save them with the extension `.txt`.

## Saving a source code file

Before you can compile and test your program, you must save it to disk. Before you can compile a program that includes a header file in a source code file, you must save the header file to disk.

To save the source code file, choose the **Save** command on the File menu (alternatively, click the **Save** tool button, or type **Ctrl+S**).

If the file was already saved earlier, Quincy writes the current version over the old one. If the source code file is a new one that you have not yet saved, Quincy opens the **Save As...** dialog box shown here.

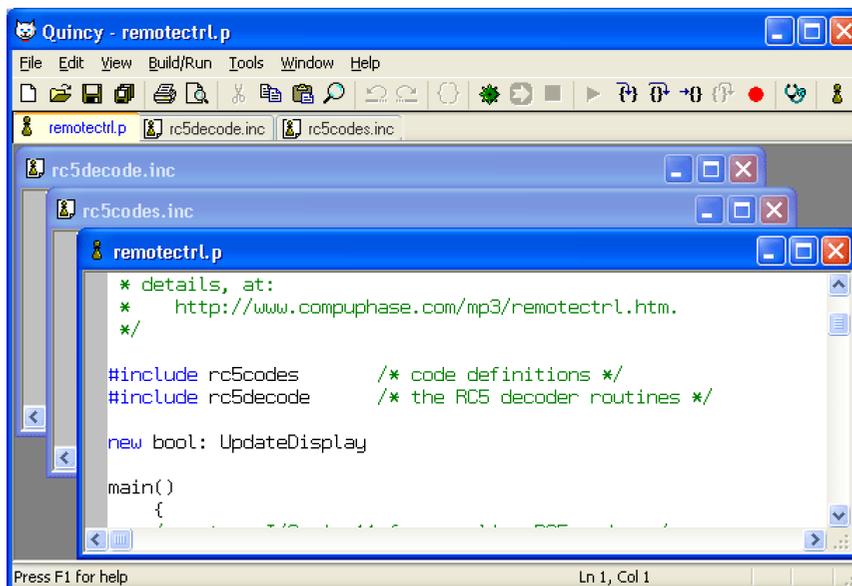


In this dialog:

1. Enter the source code file name in the File Name field. You can omit the extension. Quincy uses the extension selected in the “Save As Type” dropdown listbox.
2. Ensure that the **Save As** dialog box is positioned at the Windows folder where you want to save the source code file. If not, use the dialog box to navigate to the correct folder.
3. Click the **Save** button.

To save all the source code files loaded into the Quincy IDE, choose the **Save All** command on the File menu (or use the “Save All” tool button).

Note: Remember when you save header files to put them in the same folder as source code files that include them with the `#include "file.inc"` notation. If you save the header files in a different folder, the source code files that include them must use the `#include <file.inc>` notation, and Quincy’s options must be set to tell Quincy where to find such header files as explained in [Setting Options on page 4](#).

FIGURE 6: *Multiple documents open at the same time*

## Using Save As

You can save an existing file that was saved earlier and give it a different name by choosing the Save As command on the File menu to open the Save As . . . dialog box. See the preceding section *Saving a source code file* for details on the “Save As” dialog. The original file is not changed by this procedure.

## Working with multiple source code & include files

Quincy is a Windows Multiple Document Interface (MDI) application, which means that you can open and work with more than one source code file at a time as shown here.

If your program involves more than one file—that is, a combination one or more PAWN file plus one or more include files, it may be useful to store that set of files as a “workspace” file. To save a workspace file, use the option **Save workspace. . .** from the File menu. Quincy will pop up a Save As dialog, just like when saving a new file for the first time.

The workspace includes all files that are currently opened in the Quincy editor, plus the “build options”. If you save a workspace, the current options are saved to that workspace file as well. If you load a workspace, the options are loaded from that workspace.

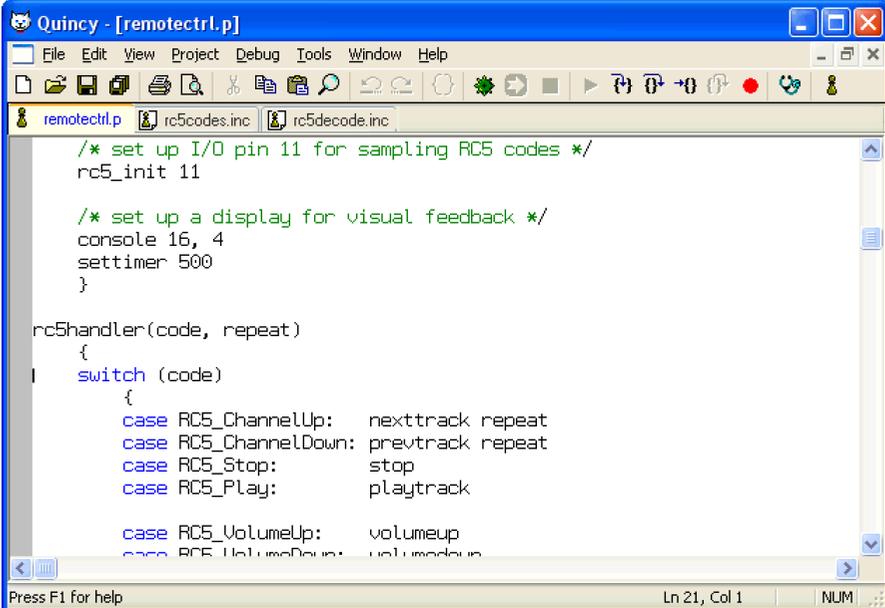
## Editor interface

---

Quincy's editor is a typical text editor similar to those found in most software development environments. If you know how to enter text into Notepad, you will know how to use Quincy. There are editor options you can set to change how Quincy displays text as you enter it. You can change the colors of the text display, how tabs are expanded, and whether the editor autoindents as you type.

### Entering Text

Enter text by typing it. Quincy does not support word wrapping; the editor window scrolls horizontally if you type past the right margin. If you have syntax highlighting selected in the editor options, the text changes color as the editor parses those text elements (comments, key words, and string literals) that should be highlighted as shown here.



The screenshot shows the Quincy text editor window titled "Quincy - [remotectl.p]". The window has a menu bar (File, Edit, View, Project, Debug, Tools, Window, Help) and a toolbar with various icons. The editor displays a C program with syntax highlighting: comments are green, keywords like 'switch' and 'case' are blue, and identifiers are black. The code is as follows:

```
remotectl.p rc5codes.inc rc5decode.inc

/* set up I/O pin 11 for sampling RC5 codes */
rc5_init 11

/* set up a display for visual feedback */
console 16, 4
settimer 500
}

rc5handler(code, repeat)
{
  switch (code)
  {
    case RC5_ChannelUp:   nexttrack repeat
    case RC5_ChannelDown: prevtrack repeat
    case RC5_Stop:       stop
    case RC5_Play:       playtrack

    case RC5_VolumeUp:   volumeup
    case RC5_VolumeDown: volumedown
  }
}
```

The status bar at the bottom indicates "Press F1 for help", "Ln 21, Col 1", and "NUM".

## The Insertion Cursor

Observe the vertical black bar in the first column position on the line that starts with the word “switch” in the illustration above (this is the 10<sup>th</sup> line of text from the top). This black vertical bar is the text insertion cursor. Although this illustration is static, when Quincy is running, the insertion cursor blinks so you can find it when you need to. When you type, the characters you type go where the cursor is positioned, and the cursor moves to the next position.

You can move the insertion cursor several ways:

1. Use the Home, End, Page Up, Page Down, and arrow keys.
2. Use those keys with the Ctrl key depressed.
3. Click in the editor window with the mouse.

The small pane in Quincy’s status bar (at the lower right of the window) displays the current line and column position of the insertion cursor. In this example, the cursor is at line 21 and column 1.

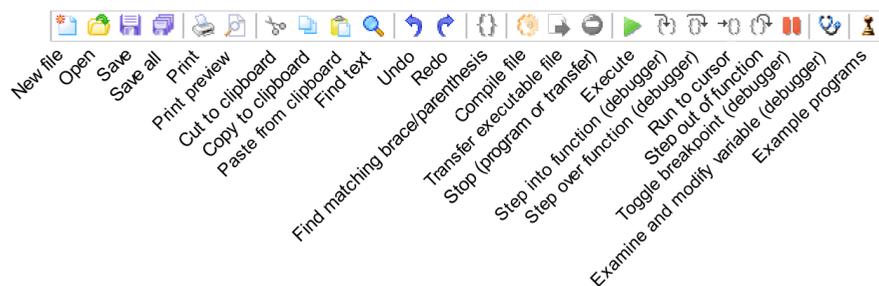
It is possible for the insertion cursor to be out of sight. If you scroll the document window horizontally or vertically with the scroll bars and the mouse, you can put the insertion cursor off screen. But it is still there. If you start typing when the insertion cursor is out of sight, Quincy changes the text’s scrolling position so that the line where the insertion cursor is positioned is in view.

Quincy’s editor does not have Insert/Overwrite modes of text entry like other editors do. Quincy is always in Insert mode, which is how editing source code is typically done.

## Brace matching

For a programming language like PAWN, where blocks of instructions are grouped between braces, it is important that those braces match correctly in pairs: for every opening brace, there must be a closing brace.

The Quincy editor provides a visual cue to help match braces, as there may be many instructions between the opening and closing brace of a group. When the insertion cursor is on a brace, the editor marks the pairing brace by drawing a rectangle around it. If the insertion cursor is on an opening brace, the editor marks the matching closing brace, and if the insertion cursor is on a closing brace, the editor marks the matching opening brace.



---

FIGURE 7: *The Quincy toolbar*

---

By pressing `Ctrl+{`, the insertion cursor jumps to the matching brace.

The Quincy editor operates the same on opening and closing parentheses. Like braces, these must match. Moving the insertion cursor on a parenthesis marks the matching parenthesis, and pressing `Ctrl+{` moves the insertion cursor to that matching parenthesis.

## Breakpoints as bookmarks

Quincy does not have an option to set bookmarks in the text for quickly jumping back and forth to specific locations. However, breakpoints may be used for this purpose while editing, because Quincy *does* have key combinations to jump to the next/previous breakpoint location.

To toggle a breakpoint on a line, type the `F9` key. The line does not need to have code on it; you can set a breakpoint in the middle of a comment. To jump to the breakpoint above the current cursor location, use `Ctrl+PageUp` and to jump to the breakpoint below the current cursor location, use `Ctrl+PageDown`

## The toolbar

Many functions of the Quincy are available through the toolbar. Figure 7 gives an overview of the various buttons of the toolbar and their functions.

## Info-tips

Quincy displays quick help in a balloon for native and public functions that it knows about. To pop up the “info-tip” balloon, move the mouse cursor over a function and wait for a second without moving the cursor. If the function is known, the balloon will pop up with a brief description of the function and its parameters.

The **Ctrl+Space** key combination will also pop up the info-tip balloon (if a match is found), but using the position of the insertion cursor as a criterion —instead of the mouse cursor.

If the function is not known, but the name of the symbol that the mouse cursor points at partially matches known functions or other symbol names in the script, the balloon will instead list those symbol names. You can use this list for text completion as well.

Finally, in a debug session, the info-tip balloon shows the current value of variables, instead of definitions of the symbols.

The “known” native and public functions for the info-tip balloon are declared in a file called *infotips.lst* in the *doc* directory of the PAWN installation. This file is a plain text file with a function signature and a brief description per line. Below is an example of a declaration from *infotips.lst*:

---

```
bool: fclose(File: handle)      Close a file
```

---

LISTING 1: *T*

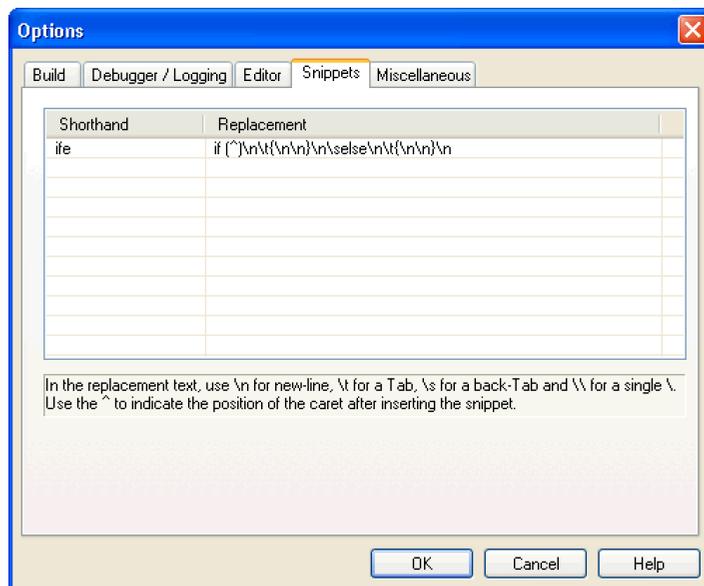
he line starts with the function’s signature. Behind the closing parentheses of the signature is the summary of the function’s purpose.

## Code snippets

To reduce the amount of typing, you can create code snippets for typical sequences of characters. To create a code snippet, select the option **Tools** from the menu and then **Options**. . . In the options dialog, select the tab **Snippets**.

For each snippet, type in a shorthand and then the replacement text. The goal is, of course, that the shorthand is short and easy to type, as well as easy to memorize.

Once you entered the code snippets, you can start using them in the editor. To expand a snippet, type the shorthand and then press the **Tab** key. For example, if

FIGURE 8: *Snippets dialog*

you have the entered the snippet as in figure 8, typing `ife` followed by a `Tab` would expand to:

```
if (  
    {  
}  
else  
    {  
}
```

LISTING 2:

## Text completion

Text completion is similar to code snippet expansion (see above), but using a different key combination: `Ctrl+Space`. Text completion expands only a single word, which it picks from all words in the current source file and from the function names declared for the “info-tip” balloon.

If multiple words match the partial word at the left of the insertion cursor, the `Ctrl+Space` combination pops up an info-tip balloon with all possible matches;

use the arrow keys to choose a word from this list and **Enter** (or **Tab**) to confirm it. For example, if you have typed in the letters “**cl**” in the editor, and press the key combination **Ctrl+Space**, an info-tip balloon will list the functions **clrscr** and **clreol**, allowing you to pick either —possibly there are more items in the balloon, depending on the script and on the info-tips list itself (which changes per configuration of the PAWN system). On the other hand, if you typed “**clre**” followed by **Ctrl+Space** (and assuming that there are no other matches), the word will expand to **clreol** immediately, without info-tip balloon.

If the word at the left of the insertion cursor is already fully expanded and this word is a known function, the summary of this function is displayed. The **Ctrl+Space** does the same as hovering over the word with the mouse, in this case.

## Searching and replacing text

Quincy supports the common operations of searching for general text and replacing a text snippet by another snippet of text. The key combinations for *local* text search and text replacement are **Ctrl+F** and **Ctrl+H** respectively.

In addition to local text searching, Quincy supports *global* text search, which is activated with **Ctrl+Shift+F** (or through the menu). A global search for matching text in *all* script files in a particular directory. The results of global search are displayed in a “Search result” view that appears at the lower part of the Quincy application window. Double-clicking on a line in that view opens/activates the relevant script file and jumps to the location of the match.

The global search function also supports “regular expressions”, as used in programmer’s utilities like Grep. In fact, Quincy uses a Grep program to perform the global search functions.

## The symbol browser

For larger scripts, using the symbol browser may be convenient to quickly jump to functions or variable declarations, based on their names. The symbol browser displays a list of all symbols combined, plus a list per category (constants, global variables, functions).

The symbol list is updated through the report that the PAWN compiler generates. Therefore, for the symbol list to be available, the option to generate the symbol report must be turned on —see section **Build Options** on [page 4](#).

Note that after inserting or deleting lines from a source file, the symbol browser is no longer up-to-date with respect to the symbols in that file. After compiling the script, the symbol browser is updated.

## Keyboard shortcuts

---

Quincy provides the standard support for cursor positioning with the keyboard (arrow keys, PageUp, PageDown) and the mouse. Marking selections in the text and clipboard operations also follow the standard interface as used in all Microsoft Windows text editors and word processors.

### Editor keys

Tab	If a block of text is selected, the <b>Tab</b> indents the block. When no block is selected, the <b>Tab</b> key expands a code snippet.
F3	Repeat the last search action.
F9	Toggle breakpoint at the active line (set or remove the breakpoint).
Alt+M	Macro playback: play a macro previously recorded.
Alt+R	Record macro: record all keyboard and mouse actions for later playback.
Alt+S	Stop recording: stop recording the macro.
Alt+Backspace	Undo (equivalent to Ctrl+Z).
Ctrl+Space	Expands the partial word at the left of the insertion cursor (i.e. “text completion”). If multiple words match the partial word, the <b>Ctrl+Space</b> combination displays a list of the possible matches in an “info-tip” balloon. You can then select a word from this list and press <b>Enter</b> or <b>Tab</b> to confirm.
Ctrl+A	Select all: select all text in the current (source) file.
Ctrl+C	Copy: copy the marked text to the clipboard.
Ctrl+F	Find: initiate a text search (in the active file).
Ctrl+Shift+F	Find globally: initiate a text search through all files in a directory (global search).
Ctrl+H	Find & replace: initiate a text “search and replace” operation (in the active file).

Ctrl+V	Paste: insert the contents of the clipboard at the position of the insertion cursor.
Ctrl+X	Cut: copy the marked text to the clipboard, then delete it in the editor.
Ctrl+Y	Redo: undo the most recent undo action.
Ctrl+Z	Undo: undo the most recent edit action.
Ctrl+Insert	Copy (equivalent to Ctrl+C).
Ctrl+PageDown	Jump to the next breakpoint in the code editor.
Ctrl+PageUp	Jump to the previous breakpoint in the code editor.
Ctrl+{	Match brace: if the insertion cursor is at an opening or closing brace, this key combination jumps to the matching brace.
Shift+Delete	Cut (equivalent to Ctrl+X).

## Debugger keys

F8	Step into: execute the current instruction and stop at the next instruction; step <i>into</i> functions (if any).
F9	Toggle breakpoint at the active line (set or remove the breakpoint).
F10	Step over: execute the current instruction and stop at the next instruction; step <i>over</i> functions (if any).
Ctrl+E	Examine: view the value of a variable, and optionally modify it.
Ctrl+L	Log: open (or switch to) the panel with the information coming from the serial connection.
Ctrl+W	Watch: open a window to watch one or more variables.
Ctrl+F10	Step to cursor: execute all instructions until arriving at the insertion cursor position.

## Miscellaneous keys

F1	Help: display general help or help on a marked keyword (if any).
F4	Tutorial: open the dialog with the examples.
F5	Run the current script. If the script's source file is more recent than the compiled script, Quincy will first ask you whether you wish to re-build the script.
F6	Open (or close) the symbol browser. See <a href="#">page 21</a> .
F7	Build the current script. This key is only available if the source file that is active at the time is a script file. Include files or text files cannot be “built”.
F12	Save as. . .: save the active file under a new name.
Ctrl+B	Close the “Build result” view with the error log of the most recent compile action.
Ctrl+N	New file: create a new text file or source file.
Ctrl+O	Open file: open an existing source file.
Ctrl+P	Print file: print the active file.
Ctrl+S	Save the active file.
Ctrl-F6	Jump to the declaration of the symbol that is currently below the text cursor. This function relies on the symbol browser. The PAWN compiler must be set up to generate the XML reports for the symbol browser. See also the section <a href="#">The symbol browser</a> on <a href="#">page 21</a>
Ctrl+F7	Transfer: transfer the compiled program to an external computer or device, over a network or serial cable. The serial connection needs to be configured for this option: see the <a href="#">Debugger / Logging options</a> on <a href="#">page 6</a> .
Ctrl+Tab	Cycle through the source files that are currently open.

## Target host configuration file

---

You can select a “target host” in the build options (see [page 4](#)). The name selected here maps to a file, which resides in the directory where the PAWN compiler as well, and has the extension “.cfg”. The file is passed to the PAWN compiler as a configuration file and it contains preset (or default) command line options for the particular target —see the PAWN “Language Guide” for details on the configuration file.

Next to the command line options for the PAWN compiler, the configuration file may also contain instructions for the Quincy IDE. Currently, the instructions are:

<b>#runtime:</b>	whether the compiled script can be run from Quincy —this is mostly true (set to 1) for desktop configurations and usually false (0) for embedded systems (where the script is built on a PC, but run on the embedded device).
<b>#debug:</b>	what kind of debug support is enabled for the platform. For a desktop configuration, local debugging (1) is usually provided; when the script runs on a different device than the system that it is built on (e.g. embedded systems), only remote debugging (2) is available, or none at all (0). If both local and remote debugging are available, this setting is 3.
<b>#optlevel:</b>	the maximum optimization level allowed for the PAWN compiler. The optimization level used by the compiler is directly related to the instruction set that abstract machine in the target host implements. A target host that uses the Just-In-Time compiler (JIT) can only support the core instruction set, so this setting should be 1. An abstract machine that supports the full instruction set will have this setting at 3.
<b>#overlay:</b>	whether the target host supports scripts with overlays (1) or not (0).

Quincy also browses through the PAWN compiler options in the configuration file and handles the following options:

<b>-d</b>	for the default level of debugging information.
<b>-o</b>	for the (default) fixed output name.
<b>-O</b>	for the default optimization level.

# Index

---

- ◊ Names of persons (not products) are in *italics*.
- ◊ Function names, constants and compiler reserved words are in **typewriter font**.

- |   |   |
|---|---|
| <p><b>A</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Auto-indent, 9</p>   | <p><b>K</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Keyboard shortcuts, 23</p>   |
| <p><b>B</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Block indenting, 23<br/>Bookmarks, 18<br/>Brace matching, 17, 24<br/>Breakpoint, 23, 24</p>  | <p><b>L</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Linux, 1<br/>Logging, <i>See</i> Serial logging</p>  |
| <p><b>C</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Code completion, <i>See</i> Text completion<br/>Code snippets, 19, 23<br/>Comments<br/>  documentation ~, 5<br/>  multi-line ~, 9<br/>Cursor (text), <i>See</i> Insertion cursor</p> | <p><b>M</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Match braces, <i>See</i> Brace matching<br/>Match parentheses, <i>See</i> Brace matching</p>   |
| <p><b>D</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Documentation comments, 5</p>  | <p><b>O</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Optimization, 4<br/>Overlay code, 5<br/>Overwrite mode, 17</p>   |
| <p><b>F</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Find symbols, <i>See</i> Symbol browser<br/>Find text, <i>See</i> Text search &amp; replace</p>  | <p><b>P</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Parenthesis matching, <i>See</i> Brace matching</p>  |
| <p><b>G</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Global search, 21, 23<br/>GREP, 21</p>   | <p><b>R</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Remote debugging, 6<br/>Replace (text), <i>See</i> Text search &amp; replace<br/>RS232, 7</p>  |
| <p><b>I</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Indent, <i>See</i> Auto-indent or Block indent<br/>Info-tips, 19–21, 23<br/>Insertion cursor, 17, 20</p>   | <p><b>S</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Script transfer, 8, 25<br/>Search (text), <i>See</i> Text search &amp; replace<br/>Serial cable, 7<br/>Serial logging, 8, 10, 24<br/>Shorthand, 19<br/>Snippets, <i>See</i> Code snippets<br/><i>Stevens, Al</i>, ii, 1<br/>Symbol browser, 5, 21, 25<br/>Syntax highlighting, 9</p> |
| <p><b>J</b> <hr style="display: inline-block; width: 300px; vertical-align: middle; margin-right: 10px;"/> Just-In-Time compiler, 4, 5, 26</p>  |   |

**T**  
Tab size, 8  
Text completion, 19, 20, 23  
Text search & replace, 21, 23  
Toolbar, 18  
Transfer (script), 8, 25

**U**  
Undo queue size, 9  
Update check, 9

**W**  
Word wrap, 16  
Workspaces, 11, 14