

SimpleIDE User Guide



Table of Contents

| | | | |
|----------------------------------------------------|----------|------------------------------------------------|-----------|
| What's New? | 2 | Tools Menu | 15 |
| SimpleIDE v1-0-0 to v1-1-0 | 2 | Button Bar..... | 15 |
| SimpleIDE v0-9-45 to v1-0-0 | 2 | Project Manager | 15 |
| Overview | 3 | File Manager Tab | 16 |
| Features | 3 | Project File Types..... | 16 |
| Software Requirements and Installation..... | 4 | Project Options | 17 |
| Downloads and Installation Instructions | 4 | Board Type..... | 17 |
| USB Drivers | 4 | Compiler Type..... | 18 |
| First Run | 4 | Memory Model..... | 18 |
| Create Workspace..... | 4 | Compiler Options | 19 |
| Start-up "About" Window..... | 4 | Linker Options..... | 19 |
| SimpleIDE Properties | 5 | SimpleIDE Terminal | 20 |
| Initial View and First Program..... | 6 | Features | 20 |
| Workspace and IDE Controls | 7 | Options | 21 |
| File Menu | 7 | Simple Libraries | 22 |
| Project Menu | 8 | How to Add a Simple Library to a Project | 22 |
| Edit Menu | 8 | Alternative | 22 |
| Tools Menu | 9 | How to Create a Simple Library | 24 |
| Program Menu..... | 11 | Simple Library Directory Structure | 25 |
| Help Menu | 12 | Recommended Features..... | 25 |
| Button Bar..... | 12 | Board Types..... | 26 |
| Port Dropdown..... | 12 | Special Clock Boards | 26 |
| Status Bar | 13 | Basic Board Types | 27 |
| Build Status | 13 | Configuration Files | 27 |
| Project View (advanced users)..... | 14 | Configuration Variable Patching | 27 |
| File Menu | 14 | Support | 29 |
| Project Menu..... | 14 | Revision History..... | 29 |

What's New?

If you have used a previous version of SimpleIDE, this section explains what differences to expect with the new version of the software.

SimpleIDE v1-0-0 v1-1-0

- Replaced loader subsystem (Propeller-Load) with new loader subsystem (PropLoader).
- Added wireless (Wi-Fi) support for programming/debugging via Parallax Wi-Fi Module (#32420). Wireless Propellers appear in port field when available; re-namable via Tools > Rename Port.
- Enhanced download speed (by 6x) for both wired (USB) and wireless (Wi-Fi) connections (requires on-board 5 MHz crystal).
- Mac installer now includes FTDI Driver and requires system restart as needed for proper FTDI driver operation.
- SimpleIDE relies on persistent storage of properties in all OSes. Updated SimpleIDE packages replace these properties during the SimpleIDE Library Install Workspace update.
- Improved terminal performance.
- Adjusted terminal to wrap to page by default.
- Fixed Find/Replace to include the first character of the search term it finds.
- Sets focus to editor after closing Find/Replace dialog to enable quicker manual replacements.
- Increased contrast on "found" item (in Find/Replace operations).
- Allows .spin object to be added to projects for library creation.
- Simplified memory model list to exclude XMM. Existing XMM applications may be compiled and downloaded with previous versions of SimpleIDE and Propeller GCC.

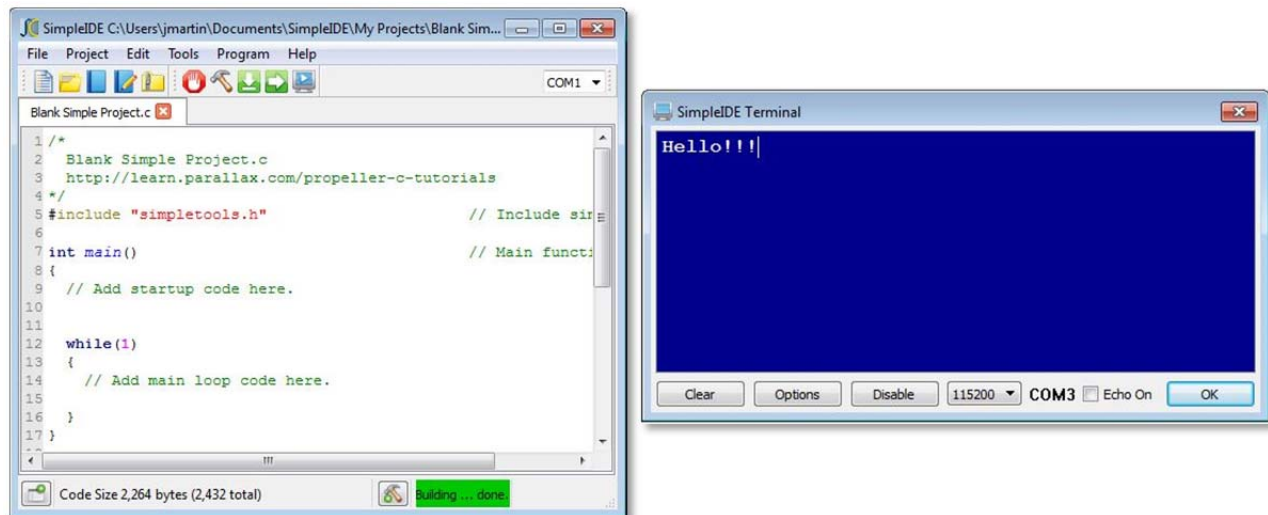
SimpleIDE v0-9-45 v1-0-0

- Enhanced COM port list to automatically update with virtual COM port events.
- Added Help > Build Error Rescue feature to easily copy build status for sharing with others.
- Added Tools > Update Workspace feature to simplify process of updating the user's workspace with library sets release later.
- Enhanced to replace missing Workspace subfolders from original set.
- Context-sensitive help added for function names and libraries.
- Added auto-main project maker for .c/.cpp files having main but no associated .side file.
- Improved Linux and OS X installation experience.
- Added USB Driver to installer where appropriate per platform.
- Made the Build Status clear upon project creation and project loading.
- Added features to lessen confusion when compiling with a non-project tab visible.
- Enhanced load options to reopen terminal after programming.
- Made the Find dialog's Find button active by default.
- Updated project manager to allow right-click shortcut menu even in blank area of view.
- Removed "Welcome to Simple View" message. Simple View is the default and is recommended for educational uses. Project View is now available only after setting the "Projects" option in the Tools > Properties > General tab.
- Removed the legacy Add Library toolbar button from Simple View.
- Removed dependency for libquazip.
- Changed Mac shortcuts from F8...F11 to be Alt+F8...Alt+F11 because of conflict on Mac.
- Fixed bug allowing leading/trailing whitespace in filenames that would break project compilation.
- Fixed bug preventing linked file types in subfolders from showing assembly.
- Fixed bug in terminal positioning feature.
- Included Simple Library updates as of 8/22/2014.

Overview

SimpleIDE is a multi-platform, open source, internationalized code development environment for the multicore Propeller microcontroller. SimpleIDE supports the C, C++, and Propeller Assembly (PASM) programming languages. It comes packaged with the PropGCC compiler for C and C++ and the OpenSpin compiler for Propeller Assembly (PASM).

SimpleIDE has a central workspace to edit code and an integrated serial terminal for exchanging information between user and Propeller microcontroller.



Features

- Menu Bar with File, Project, Edit, Tools, Program, and Help menus
- Button Bar with frequently used Menu Bar operations and COM port selector
- Streamlined New/Open/Save/Save As Project dialogs
- One-step Zip Project enables sharing with other users and the community
 - Unzipped projects are portable and may be copied to other drives or computers
- Optional Project Manager pane with right-click menu for project file settings, and Project Options, Compiler, and Linker settings tabs
 - Project Options tab enables selection of Board Type, Compiler Type, Propeller GCC Memory Model and Optimization level
- Optional Build Status pane for viewing compiler messages and build progress
- Tabbed text editor with configurable syntax highlighting
- User-selectable font size and family
- Status bar shows compile size, summary messages, and progress bar
- Available for Mac OS X, Windows, and Linux
- Multiple language interface and code support in comments and strings

Software Requirements and Installation

SimpleIDE has the following system requirements:

- Windows 7/8/8.1/10, 64-bit
- Mac OS X 10.9-10.12, 64-bit
- Linux Debian, Mint, Ubuntu 32/64-bit (.deb binary)
- 600 MB available drive space
- Available USB 2/3 port or PAB WX (#32912) w/Wi-Fi Module Installed (#32420) for programming
- Internet access for obtaining the SimpleIDE software and web tutorials

Downloads and Installation Instructions

If you wish to try Parallax Propeller C tutorials, step-by-step installation instructions can be found here: <http://learn.parallax.com/propeller-c-set-simpleide>. Links to the Propeller C Tutorials appear in the sidebar navigation.

USB Drivers

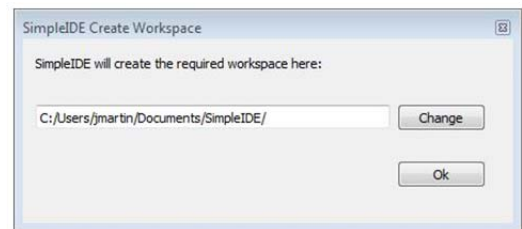
The installer package includes the USB drivers that must be installed before connecting Propeller hardware; however, you may also download them separately from here:

- Windows: <http://www.parallax.com/usbdrivers>
- Mac and Linux: <https://www.parallax.com/downloads/mac-ftdi-usb-driver>

First Run

Create Workspace

Upon first startup, SimpleIDE will prompt you to create a SimpleIDE Workspace where your libraries and project files will be stored. Just click the OK button to select the default path `~/Documents/SimpleIDE` and it will be automatically created for you.



Start-up "About" Window

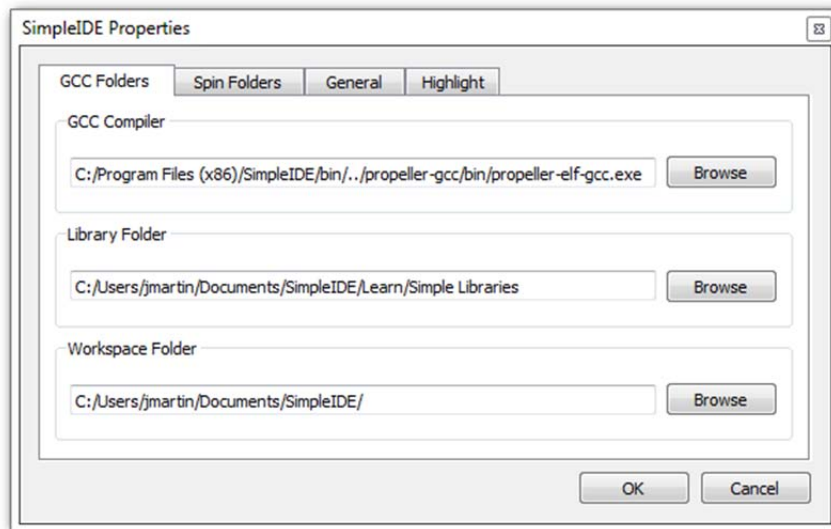
The SimpleIDE About window appears after the workspace is created. This window shows the software version and provides links to official SimpleIDE resources.

If the "Show this window at startup" box is checked, the window will appear on every startup. Clear the checkbox to disable showing this window at every startup.



SimpleIDE Properties

This dialog will normally not appear upon first run, but if it does, it's because there's an error in the critical resource paths. Typically an error will show up as one or more of the fields being blank.



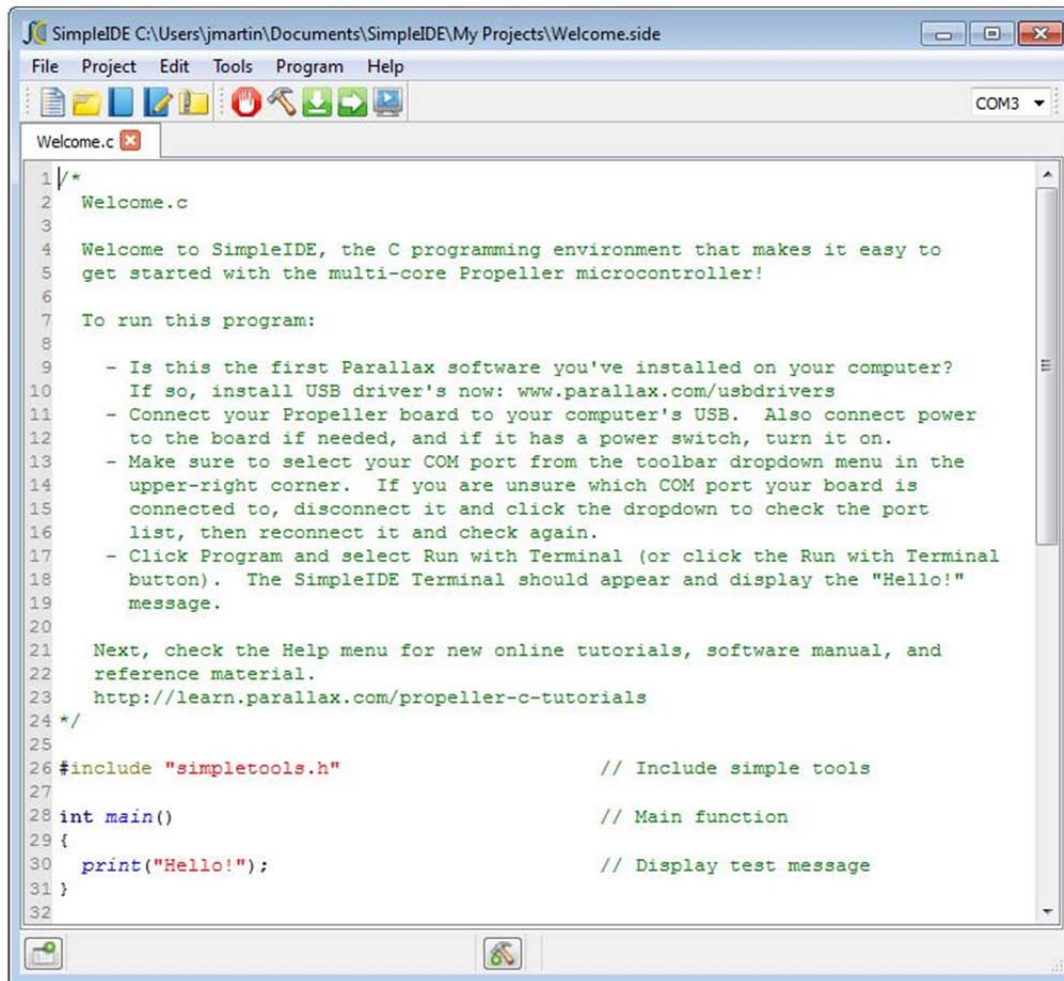
In the unlikely event that the dialog shows up, use the Browse buttons next to each field to correct its entry. See the Tools Menu section starting on page 9 for examples of default paths in Windows.

- GCC Folders tab: GCC Compiler field should contain the path to propeller-elf-gcc.exe.
- Spin Folders tab: Spin Compiler field should contain the path to openspin.exe.
- General tab: Loader Folder field should contain the path to the propeller-load subfolder of PropGCC (propeller-gcc).

Not that the back-slash '\' path separators commonly used in Windows are replaced by forward-slashes '/' in the IDE.

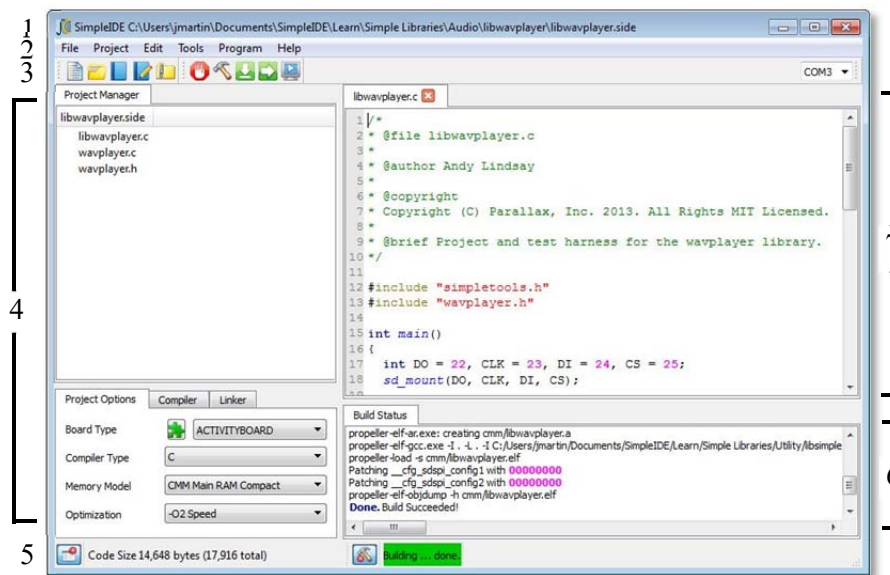
Initial View and First Program

The first time SimpleIDE runs, it will open a Welcome project in the editor. For more info about how to set up your board and run this test application, go to <http://learn.parallax.com/propeller-c-set-simpleide> and then follow the link for your operating system.



Workspace and IDE Controls

SimpleIDE's workspace consists of seven major elements shown here and listed below.



1. Title Bar: Shows project path and filename.
2. Menu Bar: File, Project, Edit, Tools, Program, and Help menu options.
3. Button Bar: Buttons for most commonly used menu items, and COM port selection dropdown.
4. Optional Project Manager: File List and Project Options tabs.
5. Status Bar: Displays code size, build information, and download progress. Buttons for Show/Hide Project Manager and Show/Hide Build Status are also available in Simple View.
6. Optional Build Status: Displays GCC compiler, Linker and download messages.
7. Editor: Text editor for C/C++/PASM code; supports multiple tabs in a single project.

Note that the software allows only one active project at a time, but may have multiple tabs open at once.

File Menu

IMPORTANT: SimpleIDE defaults to running in a mode that is recommended for all beginner-level users of the Propeller C Tutorials on the Learn website (learn.parallax.com) and during classes based on the Learn materials. The menu system described here is that of this Simple View mode. See the Project View section on page 14 for details of the advanced view.



Close All: Closes all files and projects.



Print: Prints the current document to a selected printing device.



Exit: Asks to save any unsaved files and exits the program.

Project Menu



New: Opens a dialog to select folder, project name, and project type (C or C++).



Open: Opens an existing project file (with a .side extension).



Save: Saves all of the source files in the project with their current names and locations. Note that the project is also saved automatically whenever it is compiled.



Save Project As: Save the current project with a new name to a specified folder. This will save all project settings and relative paths.



Zip: Creates a zipped archive of the project that includes all the source, header, and library archive files the project needs to compile. After the file is unzipped, the resulting folder contains a portable version of the project that can be run on other computers that have SimpleIDE installed without being dependent on libraries that the other computer might or might not have.



Close Project: Closes the project. Projects are automatically saved before they are closed.



Add Simple Library: Simplifies adding a preconfigured library to a project. This opens a dialog for selecting a folder that starts with the letters "lib." For examples, browse the subfolders of "...SimpleIDE\Learn\Simple Libraries\."

When a Simple Library folder is selected using this feature, SimpleIDE automatically adds an #include statement into the code, adds include and library paths to the Project Manager's file list, and adds -lname (where name is the characters following "lib" in the folder name) to the Linker tab's Other Linker Options field in the Project Manager. See How to Add a Simple Library to a Project on page 22 for more information.



Add Tab to Project: Creates a new file and adds it as a file link in the Project Manager. Make sure to choose the file type from the Files of Type dropdown menu. File type options are: .c, .cpp, .spin, .h, .cgc, and any file (*).



Open Tab to Project: Opens an existing file and adds it to the project. This supports the same file types as Add Tab to Project.

(Previous projects list): Lists the last 10 opened projects.

Edit Menu



Copy: Copies selected editor text to the clipboard.



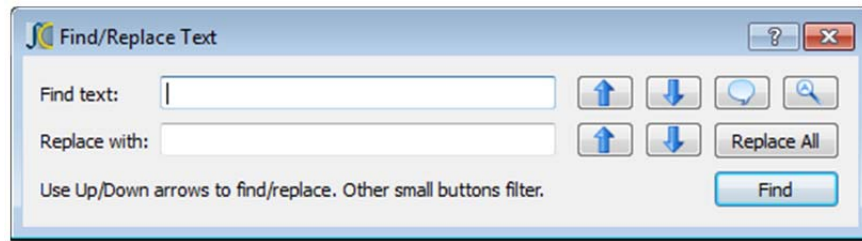
Cut: Copies selected editor text to the clipboard and deletes the selected text.



Paste: Pastes text from the clipboard to the editor at the current cursor position.



Find and Replace: Opens a dialog window that allows searching and replacing text in the editor.



Find text: When a word is entered in this field, the tool will try to find it in the editor. To find more instances of the word, click the Find Previous or Find Next buttons to the right of this field.

Replace with: When a word is entered in this field, it will be used to replace the word found from the find operation when either the Replace and Find Previous or Replace and Find Next button to the right of this field is clicked.



Find Previous or Replace and Find Previous: Finds the previous word, or replaces the current word then finds the previous word.



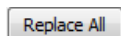
Find Next or Replace and Find Next: Finds the next word, or replaces the current word then finds the next word.



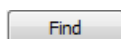
Whole Word: Finds only whole words rather than portions of words that match the Find text field's content.



Case Sensitive: Finds only words that match the case of the Find text field's contents.



Replace All: Replaces all instances of the word in the Find text field with the word in the Replace with field.



Find: Finds the next or previous word in the Find text field.



Redo: Reverses the last undo operation.



Undo: Undoes the last edit.

Tools Menu



Next Tab: Has the same effect as clicking the tab to the right of the one that is currently active in the editor pane.



Font: Allows selecting of an editor font.



Bigger Font: Increases the editor font by 20%.



Smaller Font: Decreases the editor font by 20%.



Rename Port: If a Wi-Fi device is selected in the port field, this menu item allows you to rename it to fit your needs. The name is stored in the Wi-Fi device itself for use in future sessions.



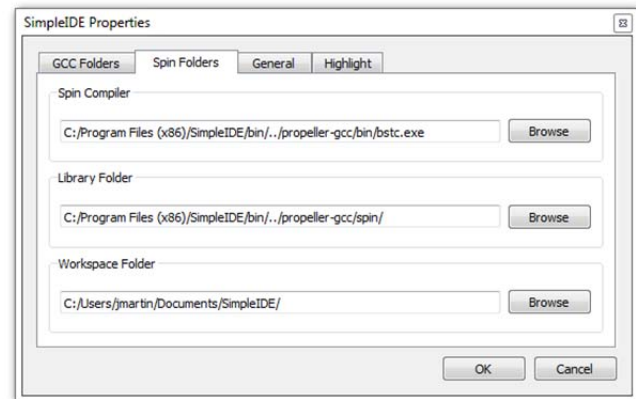
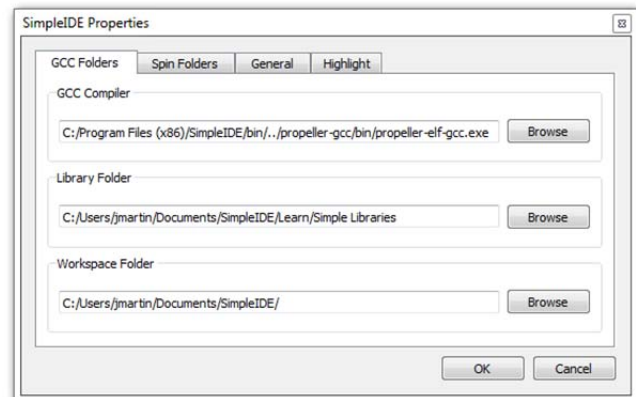
Update Workspace: Update user workspace folders to include contents of a released archive such as the Learn Library Folder from the Learn website (learn.parallax.com).



Properties: Open SimpleIDE Properties.

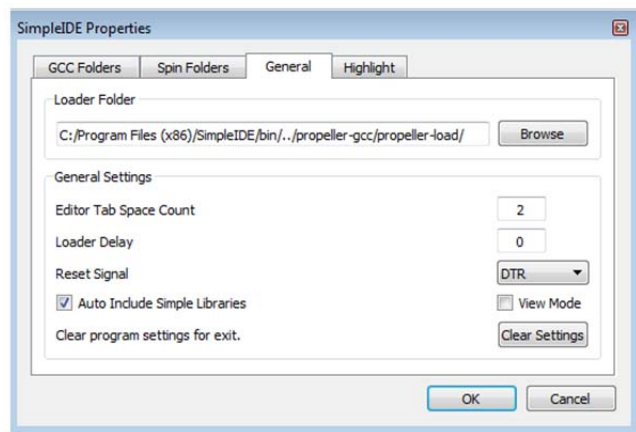
The SimpleIDE Properties dialog window allows changing of key Folders, General properties, and Highlights. The controls are consistent across operating systems and may be changed while the IDE is running to allow using another compiler directory.

GCC Folders and Spin Folders: With a default installation, the fields in the Folders tabs will be set similar to what is shown below. If the window reopens after clicking OK, it means the critical "Compiler" or "Loader Folder" information is not correctly set; use the Browse button to correct the entry. See This dialog will normally not appear upon first run, but if it does, it's because there's an error in the critical resource paths. Typically an error will show up as one or more of the fields being blank. , page 5 for more info.



General Tab: Some General property details may need to be changed for different boards.

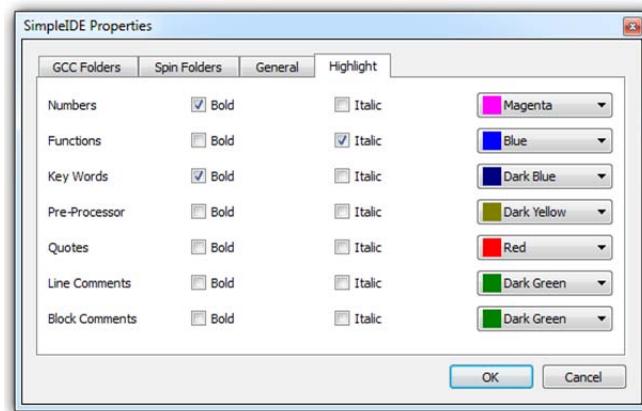
- **Loader folder:** Board type configuration files are located in the Loader folder. The contents of this folder affect the Project Options Board Type field in the Project Manager.
- **Editor Tab Space Count:** Set the number of spaces to be inserted when the Tab key is pressed. The editor will convert "tabs" to spaces. If existing source code has hard tabs in it, SimpleIDE will not touch the tabs unless the user changes them.
- **Loader Delay:** Specifies the delay between the reset signal and the start of serial transmission delivered to the Propeller. A value of 0 specifies



the recommended delay.

- **Reset Signal:** Options are DTR (default), RTS, and CFG. Some USB serial devices do not have DTR for controlling Propeller reset and should use RTS instead. The CFG option chooses the reset signal type specified in the board's configuration file.
- **Auto Include Simple Libraries:** (default) Enables automatic inclusion of Simple Libraries in projects that use an `#include <...>` statement that references them; without requiring additional -I and -L settings in Project Manager.
- **View Mode:** Enables Set Project / Simple View mode toggle (in Tools menu).
- **Clear program settings for exit:** Clears the General tab settings to restore defaults upon the next startup.

Highlight Tab: Editor syntax color and bold/italic settings can be changed here. Only a select set of system colors are available.



Program Menu



Run with Terminal (F8): Build, load, and run program on Propeller RAM. It automatically opens a serial port console terminal window.



Build Project (F9): Build program only. Build status information (such as success, or compiler error information) will appear in the Build Status pane.



Load RAM & Run (F10): Build, load, and run program on Propeller RAM.



Load EEPROM & Run (F11): Build and load program into Propeller EEPROM, then run.



Stop Build or Loader: Stop the current project build or Propeller loader operations.



File to SD Card: If the development board includes an SD card (its board type has SDLOAD or SDXMMC in its name), this button can be used to send any file to the SD Card.



Open Terminal: Toggle the display of the SimpleIDE Terminal. Press to show and connect the terminal window to the selected port. If pressed (dark square appears around the button), the

terminal is visible and connected and can be hidden and disconnected by pressing the button again.



Reset Port: This red “power switch” button allows resetting of the board, which causes the Propeller to grab the most recent program image from its EEPROM and start running it.

Help Menu

This menu provides links to the documentation for SimpleIDE, Propeller C, and Propeller GCC.

TIP: In addition to the Help menu resources, there is context-sensitive help in the source editor. In the source editor, hovering the mouse over a library function name, or over an include statement’s filename, and pressing the F1 key will bring up documentation about that item.



SimpleIDE User Guide (PDF): Opens this document.



Propeller C Tutorials (Online): Opens <http://learn.parallax.com/propeller-c-tutorials>.



Simple Library Reference: Opens the Simple Library documentation.



PropGCC Reference (Online): Opens <http://www.parallax.com/propeller-gcc>.



Build Error Rescue: Opens the contents of the Build Status pane into a dialog where it can be easily copied and shared with others (such as with Propeller Forum members) who can help resolve project build errors.



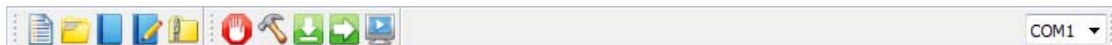
About: Shows the About window with details about SimpleIDE.



Credits: Shows links to third-party information and translation credits. License texts are distributed with the SimpleIDE package. Translations are greatly appreciated.

Button Bar

The button bar has buttons for the most frequently used menu items. Point and hover the mouse over each button to see the function it will activate when pressed. NOTE: The button selection will be different if using the advanced Project View mode; not recommended for beginners. See the Project View section on page 14.



To add additional buttons to Simple View’s button bar, simply right-click the button bar and select a button group to add. The change will last until the window is closed.

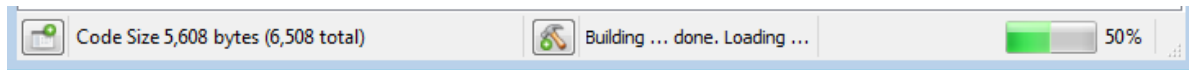
Port Dropdown

The rectangular port dropdown field on the right side of the button bar allows selection of the desired Propeller communication port. The list always shows the available ports whether or not there are any Propellers connected to the computer. When a USB-based Propeller development device is connected, this port field will automatically open to display the new selected port.

To select a Wi-Fi-based port, select the port dropdown field to see the current list of available ports and select the desired one. For proper operation, the Wi-Fi-based development board must have a 5 MHz crystal installed.

Status Bar

The Status Bar shows status information like code size, program build, and load progress.



If the Show/Hide Project Manager and Show/Hide Build status buttons are visible, clicking them will toggle the corresponding pane in the workspace area above the status bar.

The "Code Size..." notification indicates the program image size; (program + RAM size in parentheses).

The "Building..." notification indicates build and download status. A green background indicates a successful operation; a red background indicates failure. The Build Status pane automatically opens, as needed.

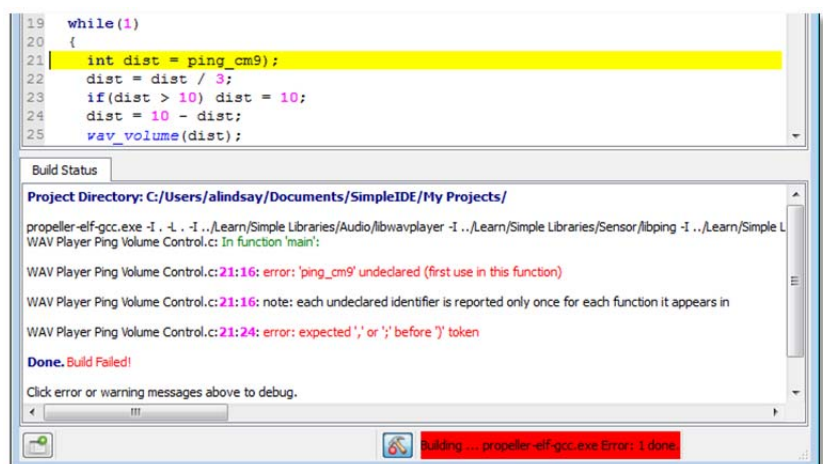
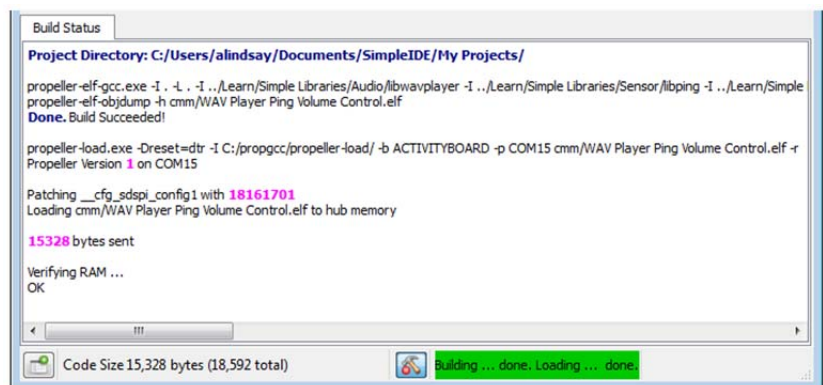
The download status appears as a progress bar and percentage complete indication at the right side of the status bar.

Build Status

The Build Status pane displays detailed reports from programs that cover the processes for code compilation and transfer to the Propeller system.

The project directory is displayed first followed by command strings passed to compiler and its report back, "Done. Build Succeeded!" It then shows the command string for the loader, the status of the program image transfer, and verification that the program was properly transferred to RAM/EEPROM.

If there is a compiler or download error, the Build Status pane will report it and will highlight the first detected error. In the example here, an opening parenthesis is missing on line 21.



Project View (advanced users)

TIP: Beginners may skip this section and move on to the SimpleIDE Terminal section on page 20.

The default features of SimpleIDE (described in all other sections of this manual) are recommended for beginner-level users and classes focused on the Propeller C Tutorials from the Learn website (learn.parallax.com).

For advanced users, an additional feature called Project View can be enabled that requires more attention to project configuration; it is not recommended except for advanced users.

To enable the Project view feature: select the Tools > Properties menu, go to the General tab, select the View Mode checkbox, and click the OK button. Then select the Tools > Set Project View menu. This is a toggle option; selecting it again will switch back to Simple View.

Project View adds items to some of the menus and button bar, as described below.

File Menu



New: Creates a new file called "untitled" in the tabbed editor space.



Open: Opens an existing file into a new editor tab.



Save: Saves the text of the active editor tab to the file shown on the tab.



Save As: Saves the active editor tab's text to a filename of your choice.



Close: Closes the currently visible tabbed editor.

(Previous file names): Lists the last 10 opened files.

Project Menu



Add File Copy: Adds a copy of an existing file to the current project.



Add File Link: Adds a link to an existing file to the current project settings.



Add Include Path: Adds a path to the current project settings pointing to a folder containing header files the project will use.



Add Library Path: Adds a path to the current project settings pointing to a folder containing memory model folders, which in turn contain archived library (.a) files. For more information, see the File Manager Tab on page 16. This feature is not for including source libraries (.c/.cpp/.cogc); use Add File Copy or Add File Link for those.



Set Project: The Set Project button (F4 or Project->Set Project) makes a project use the currently visible user program. For C language projects, this needs to be the main file.

Tools Menu



Set Project View / Simple View: Toggles between Simple View and Project View workspace modes.

Button Bar

The button bar in Project View adds a few buttons as shows below. Point and hover the mouse over each button to see the function it will activate.

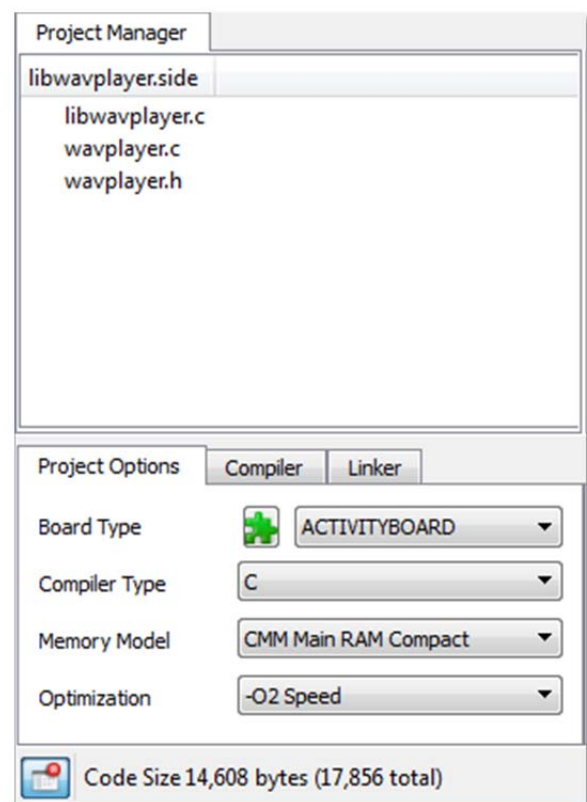


Project Manager

Projects typically consist of several files and settings that guide how the project is built and run. These are listed in the Project Manager; closed by default. If you are experimenting with Propeller C Tutorials, these project settings are already configured for you.

The minimum required file in a project is the .c file with the main function (where code execution will start). In this example, the project's name is libwavplayer.side. The .side extension is an abbreviation of SimpleIDE, and it stores the project manager settings. The file with the main function has the same name, but ends in the .c extension, and is at the top of the Project Manager's file list.

You can see how SimpleIDE uses the Project Manager to keep a list of any library, folder, and file resources the project utilizes that are not part of Propeller GCC. Libraries that are not part of Propeller GCC, or Simple Tools, have to be added to the project so that the compiler can find them. For example, you will not see stdlib listed here because that is part of Propeller GCC, so a simple `#include <stdio.h>` in the program's source code would be all that's required.



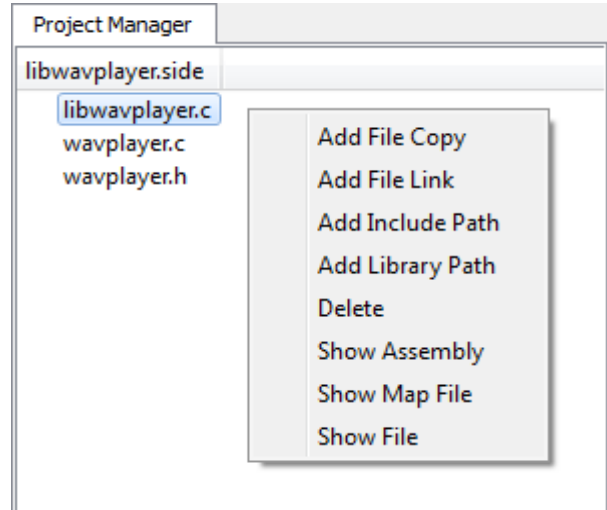
NOTES

- For C/C++, the project manager defines what is built regardless of what editor tab is active and visible in the main display.
- There is generally no issue using a main .c type file for projects that include C++ in Propeller GCC, provided the Compiler Type is set to C++.
- Although .h files are not listed by name, they must exist in the project folder or in one of the specified include paths for successful builds.

File Manager Tab

To see the contents of a file, left-click on the file name to open the file in the editor. If there are any entries starting with -I and -L, they specify paths, not files – they will not open any content in the editor. Similarly, clicking the .side file (the project file) does nothing. Controls for adding files and paths to the file list are available by right clicking one of the entries.

- **Add File Copy:** Adds a copy of an existing file to the project. If the file is outside of the project directory, the file will be copied to the directory. File names are added in alphabetical order except the main project file which must always be listed first.
- **Add File Link:** Adds a link to an existing file into the project. Unlike the above option, a linked file will not be copied to the project directory. This is useful for reusing existing code maintained outside of this project; however, a disadvantage is that the project's code is not all in same folder. Linked files are displayed in the Project Manager as the short file name with "->" and the full path and filename afterward.
- **Add Include Path:** Adds a path into the current project pointing to a folder containing header files (.h) that are not already in the project folder or tool-chain library folder. This adds an -I path to the project.
- **Add Library Path:** Adds a path into the current project pointing to a folder containing one or more memory model folders for a particular library. The memory model subfolders, with memory model names (cmm, lmm, etc.), each contain one or more .a files that have been pre-compiled for that memory model. Assuming each folder contains libname.a, -lname should be manually entered into the Linker tab's Other Linker Options field. This feature is not for including source libraries (.c/.cpp/.cogc); use Add File Copy or Add File Link for those.
- **Delete:** This option deletes a file or path from the project settings. It does not delete a file itself; rather, it just removes it from the project settings. The top file cannot be deleted since it has a special meaning to the project. The top file in the project must always have the `main()` function required by C/C++. Note that if another top file is needed, either create it or open it in Project View, and then use Project -> Set Project to set it as the top file.
- **Show Assembly:** Displays the file's Propeller GCC assembly with C source comments.
- **Show Map File:** Displays the Propeller GCC code map for a particular file.
- **Show File:** This is the same as left-clicking on a file name; it will either open that file or activate the editor tab that already contains it.



Project File Types

Source Files: These files can be added to the project manager by right clicking an existing project file and using the popup menu.

- C files (.c, .cc, .cpp) typically contain function or class method implementations.
- COG C files (.cogc) are a special type of C file that will compile to an image that will run in a cog.
- SPIN files (.spin) contain PASM that can be compiled and extracted for starting in a cog.

- GAS files (.s, .S) contain PASM-like GNU Assembly that can be compiled and extracted for starting in a cog.

Include Files: These files are also known as header files, and usually contain interface information.

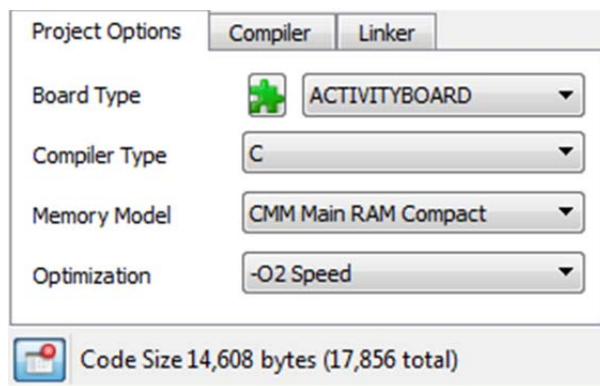
- Header files (.h) are used to define data types and declare functions that may be in libraries.
- Header files are included in C source using `#include` statements and do not need to be added to the Project Manager specifically. However, it is a good idea to add a file link because it can be used to conveniently open the header into a tab.
- If the header file is not in the same folder as the project, a path to the folder that contains the header file should be specified.
- An include path can be added by right-clicking any item in the Project Manager and selecting Add Include Path. Also, while in Project View, the Project menu's Add Include Path item can be used.
- An include path can also be added with "-I folder" in the Other Compiler Options box.

Object Files: These files are generated by the build process and are not added to the Project Manager.

- Object files (.o): Object files are always generated by the compiler.
- COG Object file (.cog): This is a generated object file created from a .cogc file.
- Dat files (.dat): This type of file is generated by Spin compilers used for making PASM COG code.

Project Options

The Project Options tab is for choosing common project settings. These options are automatically saved in the .side project file.



Board Type

This field, to the right side of the green puzzle piece, selects the target development board type. Most commercially available Propeller boards are listed here and an abbreviated list is included in Simple View. Click this dropdown to select the board you will be loading the program into. The selected board type is saved in the .side project file.

Board types with SDLOAD are special, and when selected tell the IDE that files can be loaded into the SD card. In some cases, modes like RCFAST or RCSLOW select system clock settings that have special purposes, but can be used on any board. See the Board Types section on page 26 for more info.



Reload Board Types: If board configuration (.cfg) files have been added to or deleted from Propeller GCC's Propeller Load folder (<SimpleIDE install folder>\propgcc\propeller-load\), this button will refresh the list of board names in the Board Type dropdown. If the boards.txt file (inside the Propeller Load folder noted above) has been modified, this button will update the truncated boards list shown in the Board Type dropdown while in Simple View.

Compiler Type

Two compiler types are supported:

- C
- C++

C and C++ projects are compiled by the Propeller GCC compiler that is installed with SimpleIDE.

Memory Model

The memory model options allow you to select where to store code that gets executed and data that gets accessed. There are different memory model options that utilize combinations of the Propeller microcontroller's 2 KB Cog RAM, 32 KB Main RAM (shared by all cogs) and external memories including SD and 64 or 128 KB EEPROMs built into many Propeller boards. The current version of SimpleIDE does not support every memory model type included with Propeller GCC- those supported are listed below.

Keep in mind that most memory models do not preclude other memory from being used. For example, the compact memory model (CMM) does not use SD data, but a CMM project can use libraries to read from and write to an SD card.

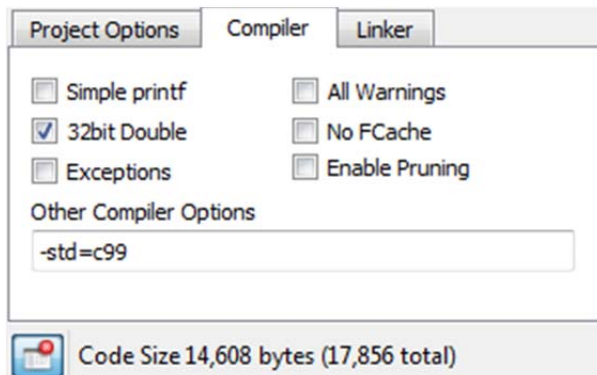
- **CMM Main RAM Compact:** The Compact Memory Model (CMM) is a size-optimized form of the LMM Main RAM model (below). This is recommended for most applications that fit entirely into the Propeller's 32 KB Main RAM. There is very little difference in performance between LMM and CMM, but the memory savings is significant.
- **LMM Main RAM:** The Large Memory Model (LMM) stores the program image and variable data in Main RAM with machine codes fetched and executed by one or more cogs.
- **COG Cog RAM:** The COG model stores both program image and data in Cog RAM. COG programs are very limited with only small amounts code and local variable data able to reside in cog memory. VGA-Pong and VGA-driver demo code are some surprising examples that will run in COG mode using Main RAM for buffering. PASM is not required.

Project examples that use a variety of memory models can be found in ...Documents\SimpleIDE\Propeller GCC Demos. Since the project stores the memory model settings, you will probably notice that most of those examples are already set to the optimal memory model for the project and target board. For example, the C-VGA demo is a COG-only program, so its memory model has been set to COG Cog RAM.

Optimization: Typically we want to optimize for size, but there are some programs that we want to optimize for speed at the cost of a larger program. Use -O2 Speed mode for speed optimizations.

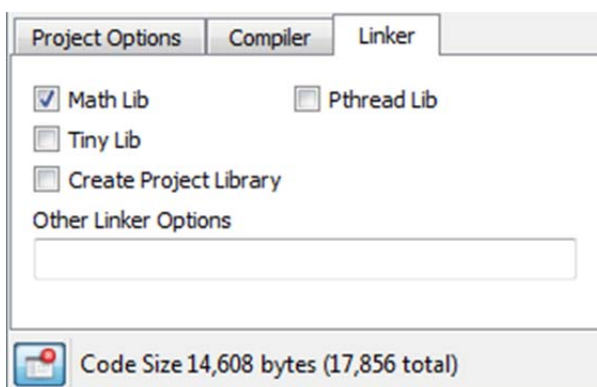
Compiler Options

Select these options as needed for a project. The defaults are fine for most projects.



- **Simple Printf:** Deprecated. See Linker tab's Tiny Lib option.
- **32-bit Doubles:** (default) Use 32-bit doubles for floating point double variables. 64-bit doubles are too big for most LMM programs.
- **All Warnings:** When checked, the compiler will generate all possible warnings for issues in code that may cause trouble.
- **No Fcache:** Check this option to prevent the compiler from using Fast Cache (Fcache). Fcache generally improves performance but it can be disabled.
- **Exceptions:** This should be enabled only for C++ programs that use try/catch exceptions. Using exceptions may cause code size to increase.
- **Enable Pruning:** Enable this option to have the compiler and linker remove unused code from the program image. This option saves the most space in projects using non-optimized libraries; Propeller GCC uses optimized libraries, so the savings are usually minimal with this option enabled.
- **Other Compiler Options:** This allows adding -D flags for programs that may need them. There are other flags that can be added here when using libraries. Use the Project Manager's Add Include Path for using prebuilt libraries. One may need -I <path-to-library-headers> for using prebuilt libraries.

Linker Options



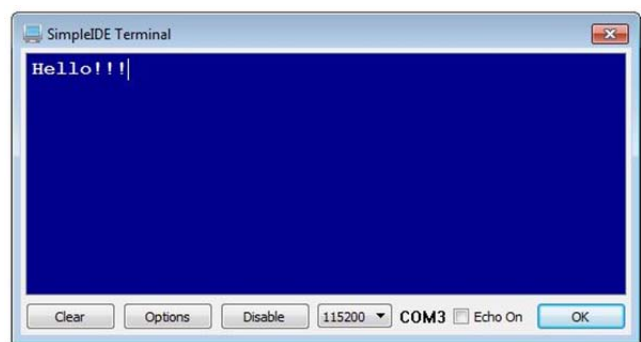
- **Math Lib:** Must be checked if using single- or double-precision floating point in the program. The program will compile without checking this, but it will not run correctly. **The Math library must be included for floating point to work.**
- **Tiny lib:** This option is deprecated; do not use it except for legacy projects. Tiny lib is an old solution meant to save code space when using `printf()`. Instead of `printf()`, use the `print()` and `printi()` functions as they save more space without the negative effects Tiny lib introduces.
- **Create Project Library:** Check this option if you are creating a library, like the Simple Tools library, where you want the archive (.a) file created during compilation.
- **Pthread Lib:** This option is only for expert users; do not enable this option without prior experience with Pthreads. This option must be checked if your project uses Pthreads for running multiple threaded programs in one cog or many cogs. The number of threads available is limited by memory. LMM programs can run M number of threads on 8 cogs. The size of M is limited only by memory available. See <http://www.parallax.com/propellergcc> for more information.
- **Other Linker Options:** This allows adding linker-specific options. For example, “-lname” may be added for using a prebuilt library. A prebuilt library has subfolders for memory models like cmm, lmm, etc., each containing a precompiled binary archive for the target memory model named libname.a. Special Linker scripts can be added here if a board does not fit a built-in memory layout.

SimpleIDE Terminal

The SimpleIDE Terminal displays the output from various text functions, such as the Standard C Library's `printf`, and `putc` functions, and Simple Library's `print` function. The terminal can also take keyboard input and transmit text to the Propeller to be received by functions like `scanf` and `getc`. Data from this window can also be shaded, copied, and pasted into other applications.

Features

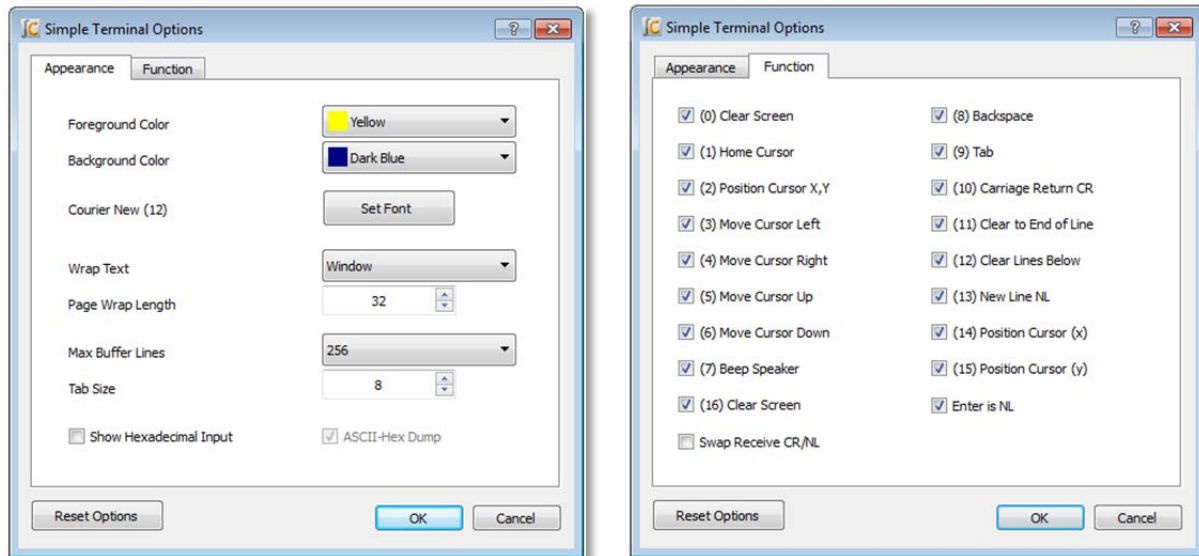
- A Clear button for clearing the terminal of any previously printed text
- An Options button for setting appearance and function
- Disable button to stop displaying incoming messages
- A baud rate selection dropdown
- A display indicating the COM Port that is/was connected
- Echo On checkbox for echoing transmitted text on the display



Options

Appearance Tab: Set color, font and tab settings. Increase the maximum buffer size setting to capture larger amounts of data.

Function Tab: Define the functions of char values that are transmitted to the terminal. For example, if checked, the value 1 sends the cursor to its top-left Home Cursor position.

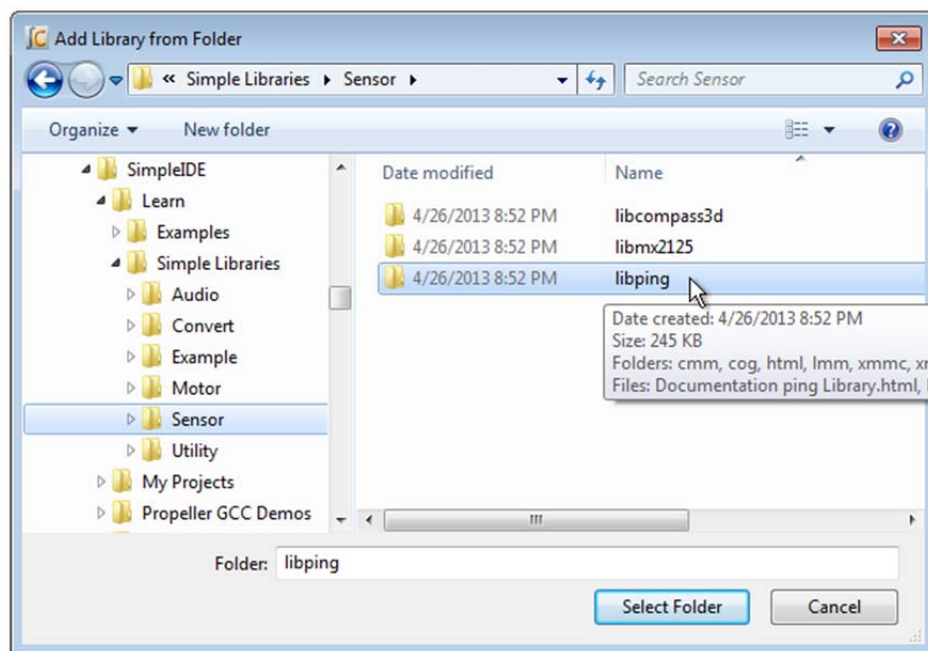


Simple Libraries

Simple Libraries are designed to be easy to add to a project with SimpleIDE. A number of Simple Libraries, including `simpletools`, are included in `...Documents/SimpleIDE/Learn/Simple Libraries/`.

How to Add a Simple Library to a Project

1. Select the Project > Add Simple Library item.
2. Browse to the library you want to add, select it, and click the Select Folder button. Note that if you accidentally double-clicked the desired folder, you can still click the Select Folder button (without selecting any subfolder first) in order to choose the desired folder.



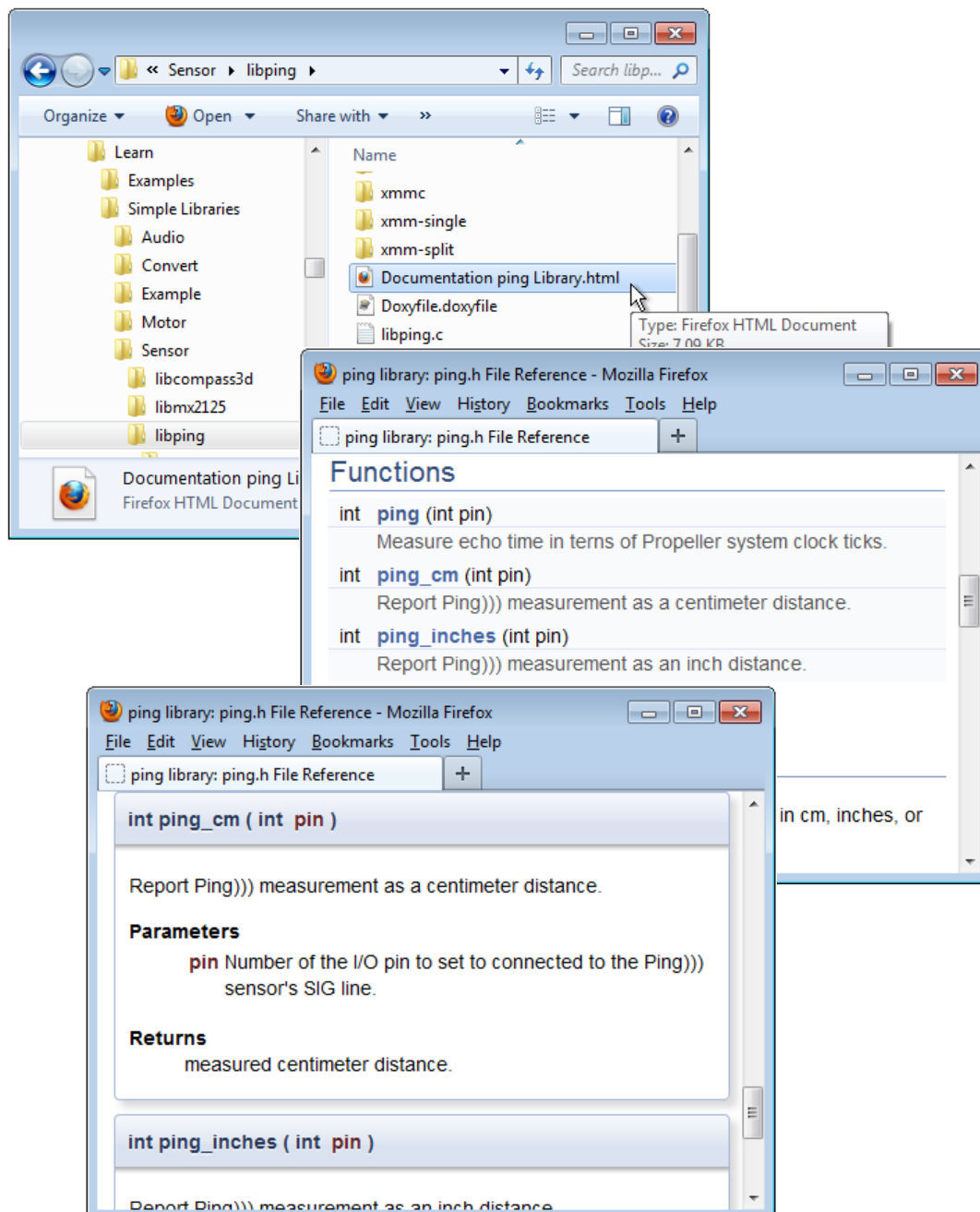
After clicking the Select Folder button, SimpleIDE will add an `#include "..."` directive for the library to your code and make the necessary additions to the Project Manager's file list and Linker tab.

Alternative

As you become familiar with Simple Library names, you can also simply insert the `#include` directive (mentioned above) into your code since the "Auto Include Simple Libraries" feature (if enabled) removes the need to adjust Project Manager's settings.

Simple Library Documentation: The Simple Library documentation lists the functions the library puts at your disposal, describes what they are for, the parameters they expect, and what they return. You can access Simple Library documentation in a number of ways.

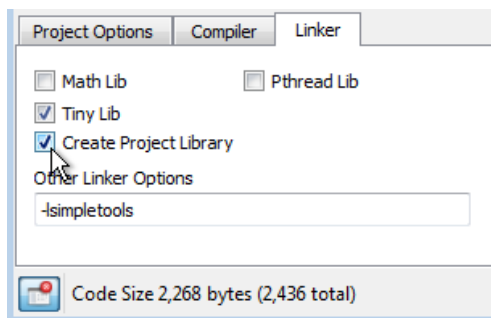
- **Use context-sensitive help:** In the source editor, hover the mouse over a library function name, or over an include statement's filename, and press the F1 key to bring up documentation about that item.
- **Use the help reference:** Select the Help > Simple Library Reference menu item to open Simple Library Documentation.
- **Use system file browser:** Use your operating system's file browser (Windows Explorer, Mac Finder, etc.) to look inside the Simple Libraries folder, and desired subfolders, and open the Documentation...Library.html file.



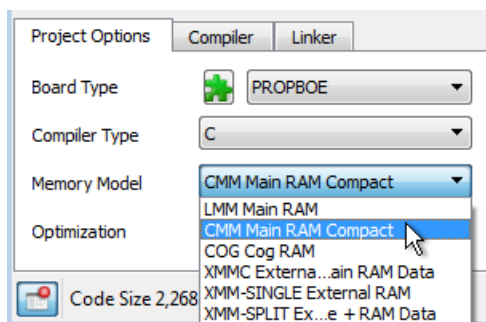
How to Create a Simple Library

To create a Simple Library (assuming it's named "name") follow these steps:

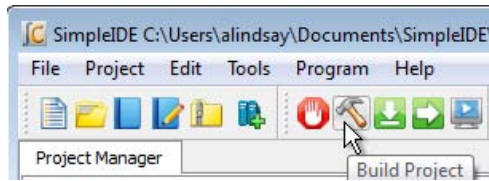
1. Select Project > New.
2. Using the New Project dialog, navigate to the desired location for your new library.
3. Click the Create New Folder icon near the upper-right of the dialog. This will add a folder called "New Folder" to the current location.
4. Rename the new folder to libname.
5. Navigate into the newly-created libname folder.
6. Change the filename (in the File name field) to libname and click the Save button.
7. Select Project > Add Tab To Project.
8. Change the filename to name, set the Save as type field to C Header File (*.h), then click the Save button.
9. If necessary, use the Project > Add Tab To Project feature to add as many .c files as needed, and try to make the names descriptive of the functions they contain. Also try to make each .c file contain as few functions as possible because the archived version will optimize out uncalled functions (.c files) if you do.
10. If you're in Simple View, click the Show Project Manager button.
11. On the Linker tab, set the Create Project Library option. This will generate a prebuilt library file (.a) in a subfolder of your new library folder every time you compile.



12. In the Project Options tab, select the desired memory model. Every time you compile the library, SimpleIDE will create an archive file (.a) in a subfolder of your library folder. For example, if CMM Main RAM Compact is the selected Memory Model, a compile of the library will create a subfolder called "cmm" with a file called libname.a inside of it. You must manually set the Memory Mode and recompile for every memory mode you want your library to support.



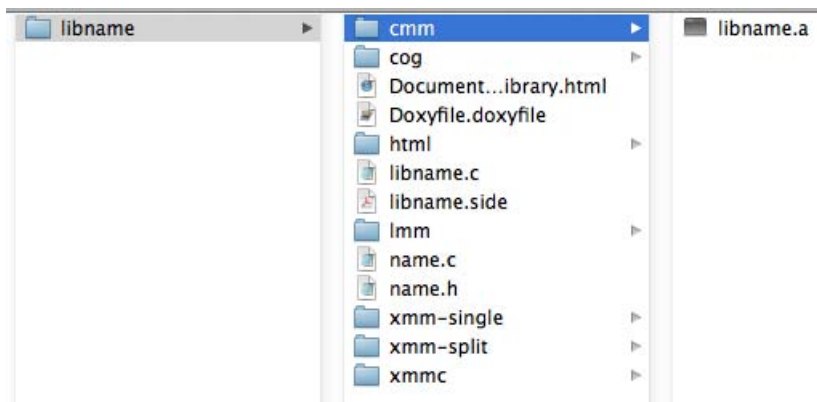
- Click the Build Project button whenever you want to compile and build the library's archive file for the currently selected Memory Model.



The selected Board Type does not affect the library's archive file (.a); it may be set to any board type while building the desired memory models.

Simple Library Directory Structure

Here is an example of how the directory structure might look for a library called "name." Note that the libname folder contains name.h and memory model subfolders, each with libname.a file. Each libname.a file is a binary archive compiled for the memory model of the folder it's in.



A Simple Library should only contain memory model subfolders for memory models it supports. For example, if the library supports only CMM and LMM, it should have cmm and lmm subfolders, but no cog subfolder. Check subfolders in ... \SimpleIDE\Learn\Simple Libraries for examples.

Recommended Features

Try to make each .c file in the library project atomic. In other words, keep one function, or a group of interdependent functions, in each .c file. When the .a file is created, it will be organized so that the compiler and linker can optimize out any functions (from separate .c files) that are not called by the project being compiled, expanding a project's image size only by the size of the functions that are called (along with any they depend on). In contrast, if the library included a single, all-encompassing .c file, with every function the library supports, then even a single function reference to the library will increase the program image size by the size of the entire library (the size of all the code for all functions within the library).

Doxygen (<http://www.doxygen.org>) is the recommended documentation tool. Try to fully document each element in the header file with Doxygen comments, and then create an html folder for your library using Doxygen. The examples in \SimpleIDE\Learn\Simple Libraries also copy name_8h.html to the libname folder and rename it Documentation name Library.html. Then the following search/replaces are performed on Documentation name Library.html with a text editor:

Search: href=""
Replace: href="html/"

Search: href="html/#"
Replace: href="#"

Board Types

SimpleIDE allows specifying board types from a simple dropdown box in Project Options. There are many board types and variations. Board type configuration files (.cfg) contain information for the loader to use when starting the program. The loader needs the main serial port, the clock mode, the clock frequency, cache driver for external memory programs, and SD card pins.


Board-specific pins can be defined in the .cfg file and "stuffed" into the Propeller program at load time so that several board types can use the same program without recompiling it. More details can be found in the propeller-load document on the PropGCC open source project website:

<https://code.google.com/p/propside>.

Special Clock Boards

If the Propeller development board does not use an 80 MHz system clock, or has an odd crystal configuration, a board type can be created to set the clock frequency (typically crystal frequency times wind-up multiplier; specified by PLLx mode). Board configuration files can be placed in the project folder, but it is best to put them in the installation's propeller-load subfolder.

Assuming <SimpleIDE Installation Folder>\propgcc is the installation folder, define a custom board as follows:

1. Copy the <SimpleIDE Installation Folder>\propgcc\propeller-load\hub.cfg file to <SimpleIDE Installation Folder>\propgcc\propeller-load\custom.cfg (or choose a name that does not have spaces).
2. Change clock frequency or clock mode in custom.cfg, and save the file.
3. If you want the board type to be available in the Project Manager's Board Type dropdown (even in Simple View), open boards.txt (from the same folder) and add your custom board's filename to the list. Boards.txt is a filter; if boards.txt is removed then all board types will be available in Simple View upon refresh.
4. Click the puzzle piece  near the Board Type field in the Project Options tab to refresh the board types list.
5. Choose CUSTOM from the board type drop down box to compile and download for the CUSTOM board you just created.

Note: RCFAST and RCSLOW board types are available in the board types list, but these should never be picked for programs that communicate through SimpleIDE Terminal because the timing is not accurate enough for serial communications.

Basic Board Types

Basic board types can work on many boards that have only a Propeller, crystal, and an EEPROM. A crystal is not necessary for RCSLOW and RCFAST board types. An EEPROM is only required if the Propeller needs to boot without the help of download from SimpleIDE.

- **GENERIC:** Supports basic hardware features of the most common Propeller boards including: Propeller Activity Board, Propeller Board of Education, PE Platform, Propeller Demo Board, and Propeller QuickStart. Although all these boards are supported, they also have their own entries in the Board Types dropdown. Each individual board type may also have extra features not included in this generic setup, so you should select the exact board type whenever it is known in advance. GENERIC settings include a 5 MHz crystal with PLL set to 16x for an 80 MHz system clock, 32 KB or larger EEPROM connected to P29 (SDA) and P28 (SCL), and 115 baud communication with a terminal via P30 (host computer Rx) and P31 (host computer Tx).
- **RCFAST:** This board type makes the Propeller microcontroller rely on its internal 12 MHz oscillator. Although it can actually boot up on any Propeller board, it is designed for applications that do not require the clock precision of an external oscillator. This mode is impractical for serial communication, servo control, precise pulse measurement, or video.
- **RCSLOW:** RCSLOW is like RCFAST except that it uses the slowest and most energy efficient clock mode in the Propeller, 20 kHz.
- **HUB:** The HUB board type specifies an 80 MHz system clock and an external 5 MHz crystal with PLL16x clock mode. Any board that has a 5 MHz crystal should work with HUB board type. Good serial communications and TV/VGA output are possible with the HUB board type and an accurate 5 MHz crystal. Some of the other board types related to HUB are ACTIVITYBOARD, C3, QUICKSTART, PEKIT, and PROPBOE.
- **SPINSTAMP:** The SPINSTAMP board type is like the HUB board type except that it relies on a 10 MHz crystal to produce an 80 MHz system clock by using PLL8x in the clock mode. Any Propeller board that has a 10 MHz crystal should work with the SPINSTAMP board type. HYDRA is a related board type.

Configuration Files

Board configuration files provide customization for Propeller GCC board hardware. A Propeller GCC program is loaded to the hardware with the propeller-load program (see <http://www.parallax.com/propellergcc/>) or, in SimpleIDE v1.1 or later, the proloader program (see <https://github.com/parallaxinc/PropLoader>). The loader will scan the board type .cfg file to use with the compiled Propeller GCC program and patch properties to the program if necessary before loading.

Configuration Variable Patching

Numeric properties found in the .cfg file can be applied to a Propeller GCC program to let the program run on boards with different hardware connections. Any variable can be defined. See <http://www.parallax.com/propellergcc/> for more information.

As a simple example, one board can have an LED on pin 15 and another board can have an LED on pin 20. The same code can run on boards using different .cfg files.

```
# file: led15.cfg
# LED pin 15 example
  clkfreq: 80000000
  clkmode: XTAL1+PLL16X
  baudrate: 115200
```

```

    rxpin: 31
    txpin: 30
    ledpin: 15

# file: led20.cfg
# LED pin 15 example
    clkfreq: 80000000
    clkmode: XTAL1+PLL16X
    baudrate: 115200
    rxpin: 31
    txpin: 30
    ledpin: 20

```

The Propeller GCC program that uses **ledpin** can look like this:

```

#include <propeller.h>
/* Config variables must be global.
 * If a variable is patched it will not have the value -1.
 */
int _cfg_ledpin = -1;
int main(void)
{
    if(_cfg_ledpin > -1) {
        DIRA |= (1 << _cfg_ledpin);
        while(1) {
            waitcnt(CLK_FREQ/2+CNT);
            OUTA |= (1 << _cfg_ledpin);
            waitcnt(CLK_FREQ/2+CNT);
        }
    }
    while(1);
    return 0;
}

```

Support

Report problems with SimpleIDE or Propeller-GCC via email to SimpleIDE@parallax.com, by adding issues to <https://github.com/parallaxinc/SimpleIDE/issues>, or by visiting the Propeller forum at <http://forums.parallax.com/>.

Revision History

For complete details, look at <https://github.com/parallaxinc/SimpleIDE> for full software revision history and source files.

- Version 1.1.0 January 23, 2017.
- Version 1.0.0 August 24, 2014.
- Version 0.9.45 November 12, 2013.
- Version 0.9.43 September 5, 2013.
- Preliminary August 6, 2013 untracked.
- Preliminary May 26, 2012 untracked.
- Preliminary Beta June 16, 2012 untracked.
- Beta April 30, 2013 untracked.
- Version 0.9.26, first released with SimpleIDE version 0.9.26.