

SimpleIDE User Guide



Table of Contents

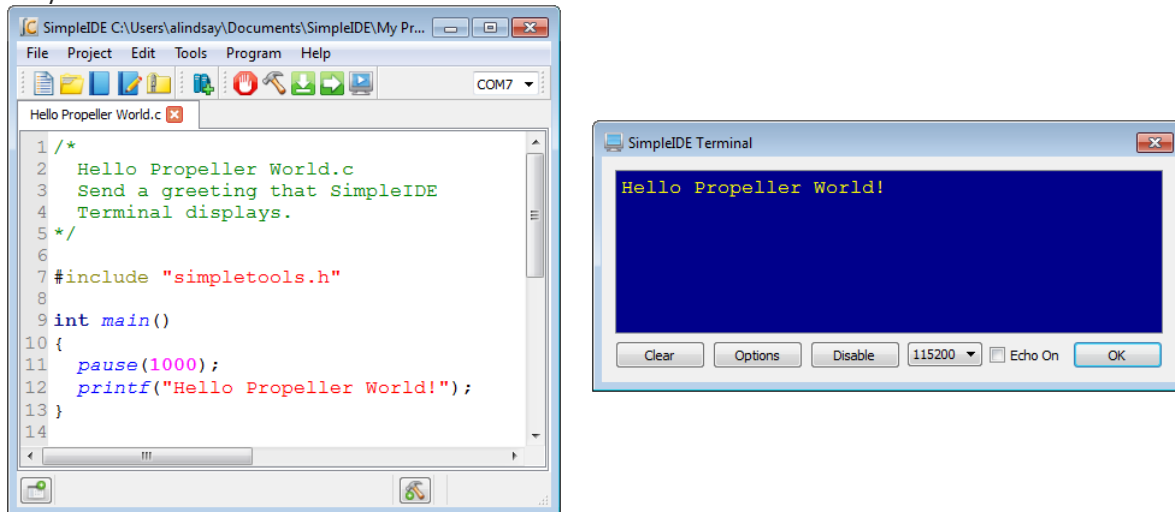
Overview	2	Memory Model.....	20
Features	3	Compiler Options	21
Software Installation.....	3	Linker Options.....	22
Downloads and Installation Instructions	3	SimpleIDE Terminal	23
USB Drivers	3	Features	23
First Run	4	Options	23
Start-up "About" Splash.....	4	Simple Libraries	25
Hint: Simple View vs. Project View	5	How to add a Simple Library to a Project	25
SimpleIDE Properties	5	How to Create a Simple Library	27
Initial View and First Program.....	6	Simple Library Directory Structure	28
Workspace and IDE Controls	7	Recommended Features.....	28
File Menu	8	Board Types.....	29
Project Menu	8	Special Clock Boards	29
Edit Menu	9	Basic Board Types	29
Tools Menu	10	EEPROM Board Types	30
Program Menu	13	External Flash Board Types	30
Serial Dropdown Menu (Button Bar)	13	External RAM Board Types	30
Help Menu	14	SDLOAD Board Types	31
Button Bar.....	14	SDXMMC Board Types	31
Status Bar	15	SimpleIDE SDLOAD and SDXMMC Attributes	31
Build Status	16	Configuration Files	32
Project Manager	17	Configuration Variable Patching	32
File Manager Tab	18	Future Improvements	33
Project File Types	19	Support	33
Project Options	19	Revision History.....	33
Board Type	20		
Compiler Type	20		

Overview

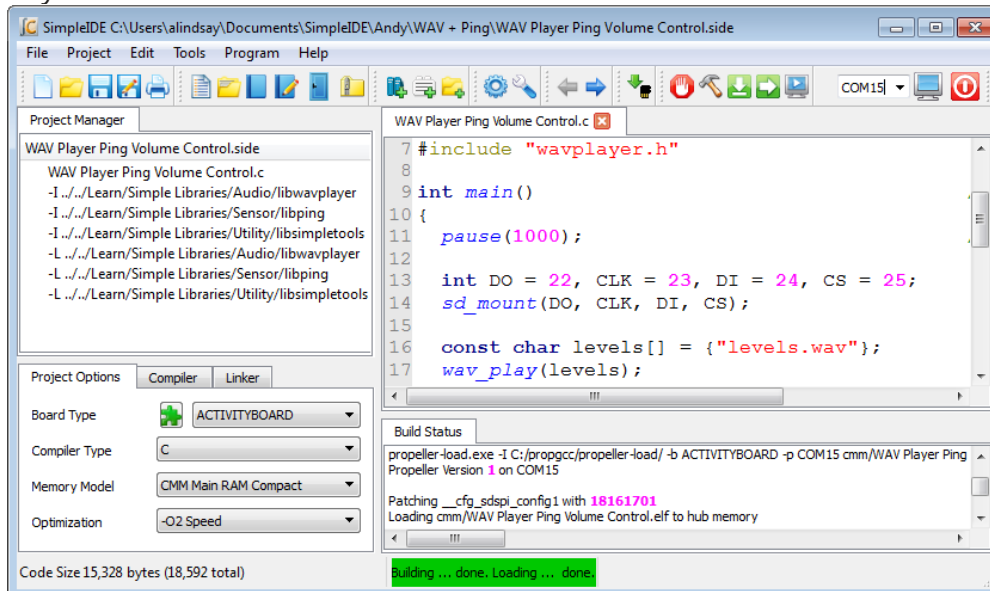
SimpleIDE is a multi-platform, open source, internationalized code development environment for the multicore Propeller microcontroller. SimpleIDE supports the C, C++, Spin and Propeller Assembly (PASM) programming languages. It comes packaged with the PropGCC compiler for C and C++ and the BSTC (Brad's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

SimpleIDE has a Workspace with two View options, Simple and Project. It also has an integrated serial terminal that can be launched into a separate window for exchanging information between user and Propeller chip.

Simple View



Project View



Features

- Two GUI options:
 - Simple View for introductory levels in the Parallax Propeller C Education Program
 - Project View (from earlier revisions) allows file-level operations for custom library and project setups
- 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 project folders are portable and may be copied to other drives or computers
- Collapsible 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
- Collapsible Build Status pane for viewing compiler messages and build progress
- Tabbed text editor with configurable syntax highlighting
- User-selectable font size and family
- Source Browser finds declarations
- Build Status shows build progress
- Status bar shows compile size, summary messages, and progress bar
- Available for Linux, Mac OSX, and Windows
- Multiple language interface and code-capable support in comments and strings

Software Installation

SimpleIDE is available for the following operating systems:

- Windows XP, 7, or 8 (install package)
- Mac OSX (install package)
- Linux setup for Debian, Fedora, Mint, Suse, Ubuntu, and Open Source build

Downloads and Installation Instructions

If you plan to try out Parallax Propeller tutorials, step-by-step installation instructions can be found here:

<http://learn.parallax.com/propeller-c-set-simpleide>

USB Drivers

USB drivers are not included in the installer package. Install drivers before connecting Propeller hardware. Download them free:

- Windows: to www.parallax.com/usbd drivers
- Mac and Linux: <http://www.ftdichip.com/Drivers/VCP.htm>

First Run

Start-up "About" Splash

At first startup, the SimpleIDE "About" window appears. This shows:

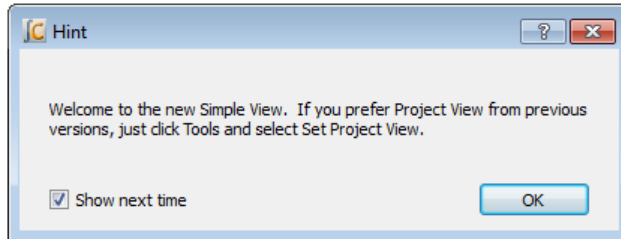
- the current version (which might be greater than the one shown below)
- a link to this user guide (or web site containing it)
- a check-box to choose showing or disabling the window at start-up
- an OK button to close the window

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



Hint: Simple View vs. Project View

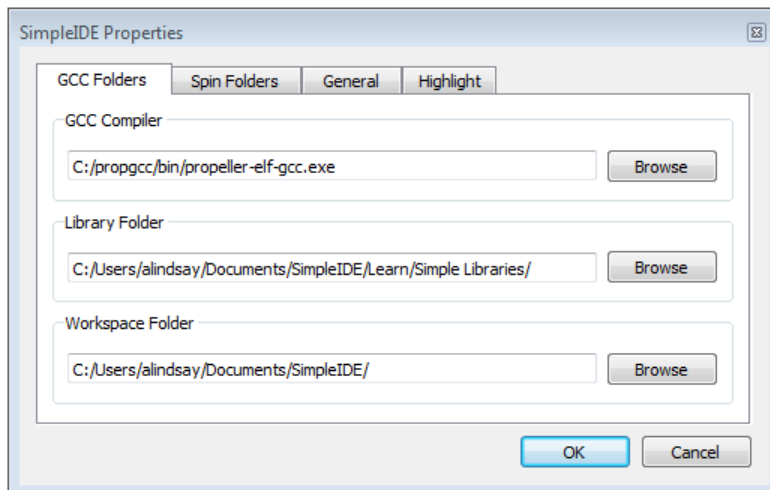
Next, a hint dialog explains that there is a new Simple View, which is the default upon installation. The original Project View, with many improvements, is still available under Tools > Set Project View. Clear the "Show next time" checkbox to prevent displaying this message at startup.



SimpleIDE Properties

At first startup, a dialog window titled SimpleIDE Properties appears. It is preconfigured with the recommended settings.

Make a note of the Workspace Folder location. This folder has subfolders for user projects, tutorial examples and demos. These settings can be changed or updated later from the Tools > Properties (F6) menu.



In most cases, the Folders tab will already have the fields properly set. The back-slash '\' folder separators used in windows are replaced by '/' in the IDE.

In the unlikely event that the window reopens after clicking OK, it means that critical "Compiler" or "Loader Folder" information is not correctly set. In this case, use the Browse buttons next to each field to correct its entry.

- GCC Folders tab: GCC Compiler field should contain path to propeller-elf-gcc.exe.
- Spin Folders tab: Spin Compiler field should contain path to bstc.exe.
- General tab: Loader folder should point to the propeller-load folder in propgcc.

See the Tools Menu section starting on page 10 for screen captures of each of these tabs with examples of default paths in windows.

Initial View and First Program

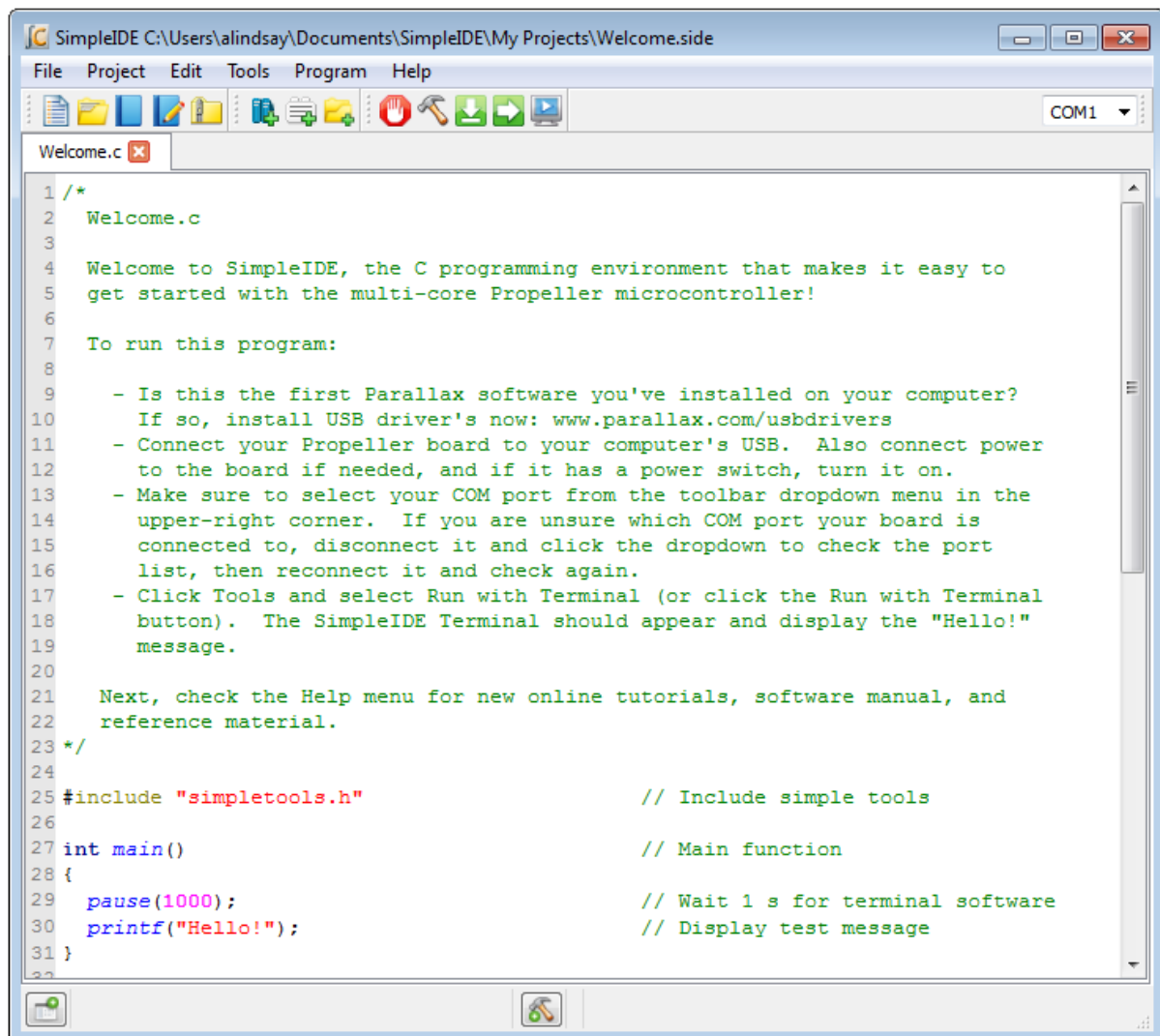
The first time SimpleIDE runs, it will open a Welcome project in Simple View. If you prefer Project View, just click Tools and select Set Project View. View settings persist, so the view selected before closing the program will still be in effect when the program is reopened.

TIP: If you want periodically view the Project Manager or Build Status, just click the Show/Hide Project Manager and Show/Hide Build status buttons in the status bar at the bottom of the IDE window.

For more info about how to set up your board and run this test application, go to:

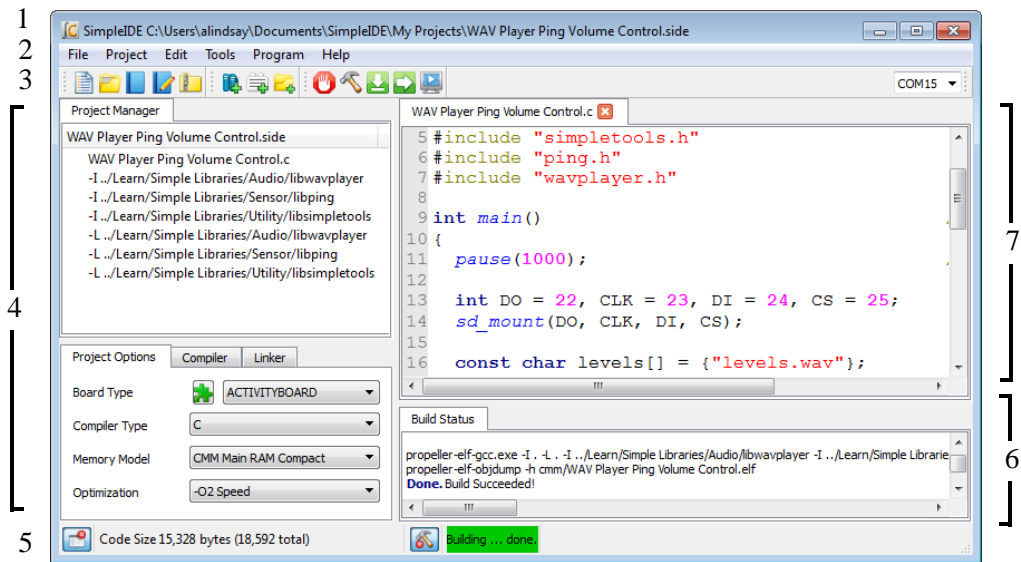
<http://learn.parallax.com/propeller-c-set-simpleide>

Then, follow the link for your operating system for instructions.



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. 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. Build Status: Displays GCC compiler, linker and download messages.
7. Editor: Text editor for C/C++/Spin code; supports multiple tabs in a single project.

Note that each instance of the window contains one active project that may have one or more tabs. If you want to work on more than one project at a time, run a second instance of SimpleIDE. In Windows you can open another instance by double-clicking a shortcut to SimpleIDE (Desktop, Start -> All Programs), or you can simply double-click the .side project file you want to work with, and it will open into a new SimpleIDE window.

Controls are available in both Simple and Project Views, unless otherwise noted.

File Menu

IMPORTANT: The first five controls here are only available in Project View, not in Simple View. Simple View only relies on the Project Menu versions of these same controls. This helps prevent confusion in beginner-level C language classes and tutorials, where all these operations have to be performed at the Project level, not the File level.



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



Open: Opens an existing file in the tabbed editor space.



Save: Saves the tab editor text to the filename in shown on the tab.



Save As: Saves the current tabbed editor text to another filename.



Close: Closes the currently visible tabbed editor.



Close All: Closes all files and projects.



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

(Previous file names): Lists the last 5 opened files.



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

Project Menu



New Project: Opens a dialog for selecting folder, project name, and project type (C, C++ or Spin).



Open Project: Opens an existing project file with extension .side



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



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



Zip: Creates a zipped archive of the project that includes all the source, header, and 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 computer might or might not have.



Add File Copy (Project View Only): Adds a copy of an existing file to the current project.



Add File Link (Project View Only): Adds a link to an existing file to the current project settings.



Add Include Path (Project View Only): Adds an include path to a folder that contains header files the project will use to the current project settings.



Add Library Path (Project View Only): Adds a path to a folder that contains memory model folders, which in turn contain archived library (.a) files. See File Manager Tab on page 18 for more info. This is not for including source libraries (.c/.cpp/.cogc). Use Add File Copy or Add File Link for those.



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



Set Project: (Project View Only): The function of the set project button (F4 or Project->Set Project) is used to make a project use the currently visible user program. For C language projects, this needs to be the main file. For Spin language projects, if you open a Spin file that is the top object file in the project, make sure to use this button while the top file is the active tab before compiling.



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

When a simple library folder is selected, SimpleIDE automatically adds a **#include** to the code, 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 Other Linker Options field under the Linker Tab in the Project Manager. See How to add a Simple Library to a Project on 25 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, .cogc, .ecog, .espin, and any file (.*)



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

(Previous projects list): Lists the last 5 opened projects.

Edit Menu



Copy: Copies selected editor text to the clipboard.

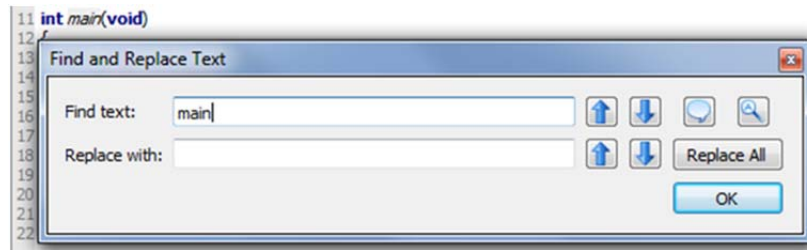


Cut: Copies selected editor text to the clipboard and deletes text.




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


 **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 Previous or Next buttons to the right of this field.

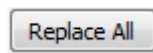
Replace with: When a word is entered in this field, it will be used to replace the word in the Find text field when either the Previous or Next button to the right of this field is clicked.

 **Previous:** finds or replaces the previous word depending on the row of the button.

 **Next:** finds or replaces the next word depending on the row of the button.

 **Whole Word:** find only whole words.


 **Case Sensitive:** find only words that match the case of the Find text field contents.


 **Replace All:** replace all instances of the word in the Find text field with the word in the Replace with field in the current editor.


 **Redo:** Undoes the last undo.


 **Undo:** Reverses the last edit.

Tools Menu

 **Set Simple View/Set Project View:** If currently in Simple View, toggle to Project View. If currently in Project View, toggle to simple view.

 **Go Back:** When not greyed, pressing this button will make the cursor jump back to the line with the function call or variable name that was browsed, see Browse Declaration below.

 **Browse Declaration:** If cursor is on a call to a user-defined function, this button will jump to the function. Likewise, if cursor is on a global variable in the code, this button will jump to the variable declaration. Library functions such as `printf` that are not in the project file-list (because they are part of PropGCC) cannot be browsed.

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

 **Font:** Allows selecting an editor font.

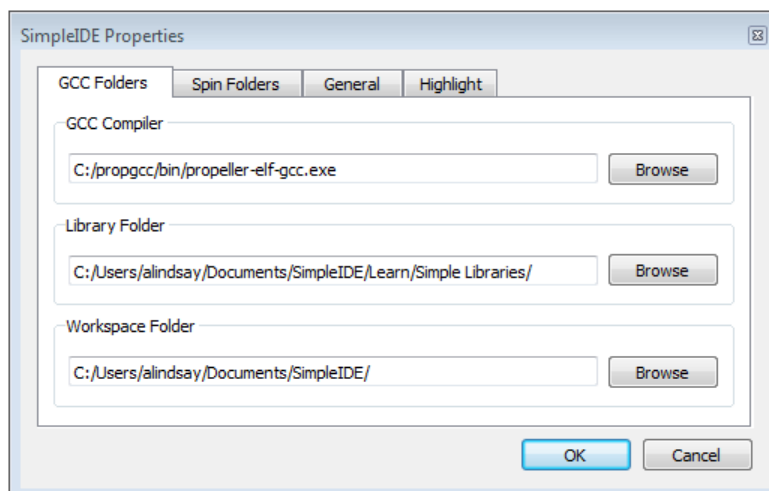
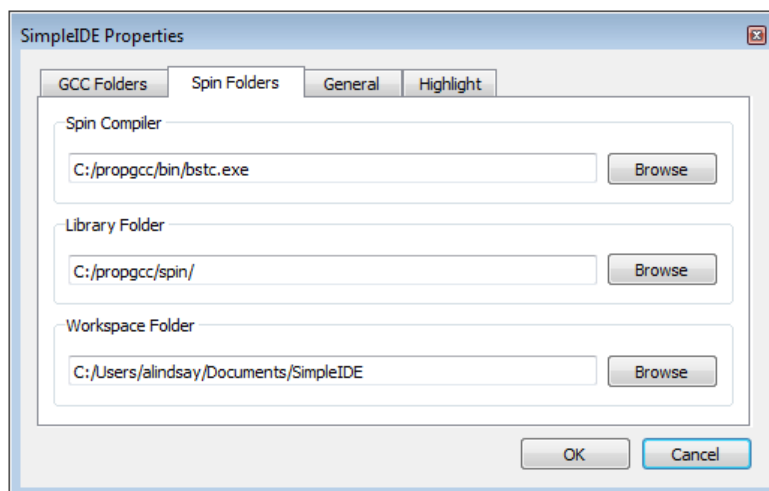
 **Bigger Font:** Increases the editor font by 20%.

 **Smaller Font:** Decreases the editor font by 20%.

 **Properties:** Open SimpleIDE Properties (F6).

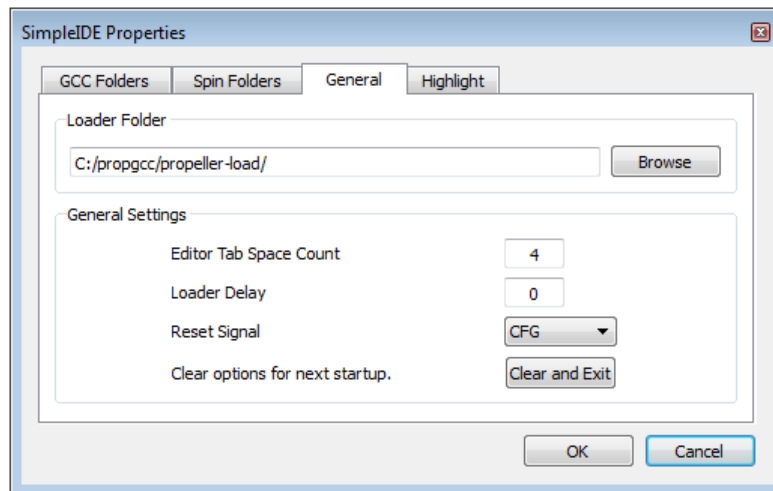
The SimpleIDE Properties dialog window allows changing key Folders, General properties, and Highlights. The controls are consistent over all operating systems and may be changed when the IDE is running to allow using another compiler directory.

GCC Folders and **Spin Folders:** With a default installation, the fields in the Folders tabs will be set as 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 SimpleIDE Properties, page 5 for more info.

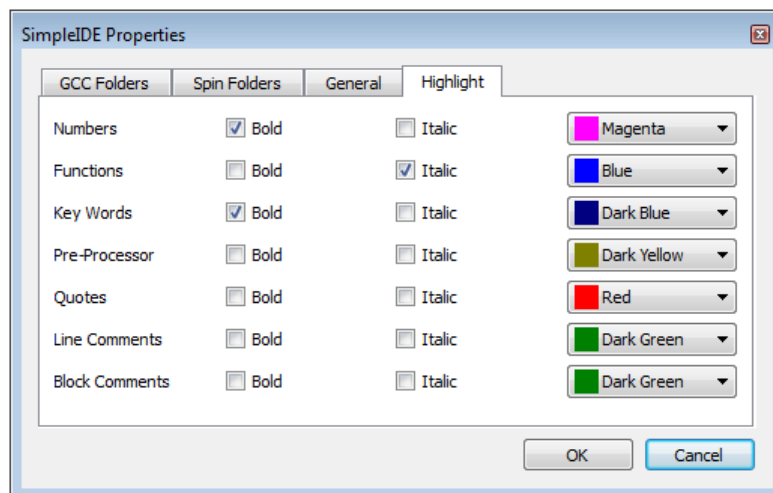


General Tab: Some general property details may need to be changed for different boards.

- **Loader folder:** Board configuration files are located in the Loader folder.
- **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.
- **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.
- **Clear options for next startup:** Clears the General tab settings to restore defaults upon the next startup.



Highlight Tab: Change editor syntax color and bold/italic settings here. At this time only a select set of system colors are available.



Program Menu



Run with Terminal (F8): Build, load, and run program on Propeller RAM (or external memory for XMM). 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 (or external flash for XMM). Board types with SDLOAD or SDXMMC in their name will have program saved to SD card first.



Burn F11): Build and load program to Propeller EEPROM (and external flash for XMM). Board types with SDLOAD or SDXMMC in their name will have program saved to SD card first.



File to SD Card: If the board-type has SDLOAD or SDXMMC in the name, this button can be used to send any file to the SD Card.

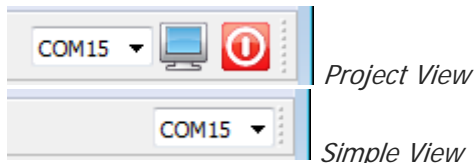


Open Terminal: This is a "display" button. Press to show and connect terminal window to the selected port. If pressed (square around the button), the port is connected and can be disconnected by pressing the button again.



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

Serial Dropdown Menu (Button Bar)



The serial dropdown menu is at the far right of the button bar. When you click it to make the port list appear, it automatically rescans the ports.

Some computers have serial ports like Bluetooth. Bluetooth serial port programming is possible, but not recommended for new users. Port type can be checked by placing the mouse cursor over the port name without clicking to see "hover help". Make sure the port type shows "USB Serial ..." or "FT232 ..." and select one of them.


SimpleIDE does not automatically detect and use the Propeller port at this time. The user must specify the port.

Help Menu


This menu provides links to SimpleIDE software documentation, online Propeller C tutorials, online Propeller GCC reference, IDE version information, and developer credits.

 **SimpleIDE Manual** (PDF): Opens this document.

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

 **PropGCC Reference** (Online): Opens www.parallax.com/propellergcc

 **About:** Shows the startup splash, SimpleIDE version number, startup disable check-box, Propeller-GCC on-line link, and the Propeller-GCC bug report support email.

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

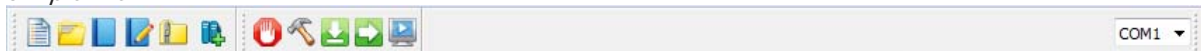
Button Bar

The button bar has buttons for the most frequently used menu items.

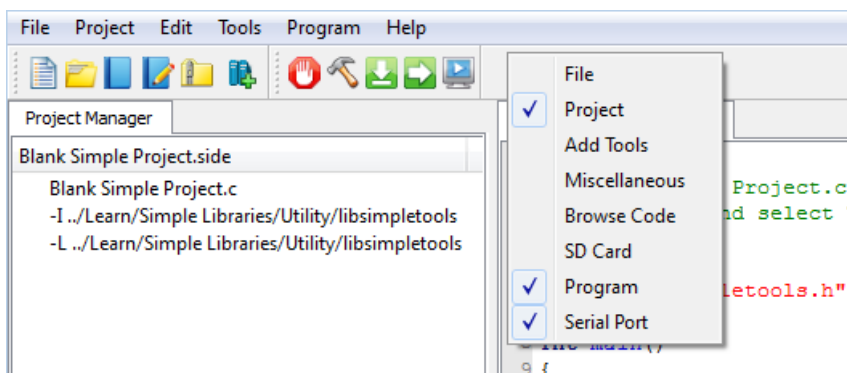
Project View



Simple View



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.




Status Bar


The Status Bar shows status information code size, program build, and load progress. The Project View of the Status Bar is shown below.

Project View



Simple View has two additional buttons: Show/Hide Project Manager and Show/Hide Build status . Those two buttons are not visible in Project View.

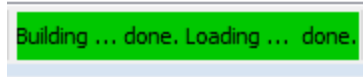
 **Show/Hide Project Manager** (Simple View Only): If Project Manager is not visible, this button causes it to expand to the left of the Editor pane. If it is visible, the button causes it to collapse.

 **Show/Hide Build Status** (Simple View Only): If Build Status is not visible, this button causes it to expand below the Editor pane. If it is visible, the button causes it to collapse.

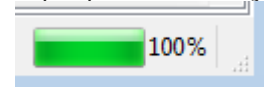
Code Size: This lists the program image size, followed by program image + RAM in parentheses.

Code Size 5,608 bytes (6,508 total)

Build Status: Summary messages display whether a build and/or download succeeded or failed. Also displays certain steps such as building... and loading... A green background indicates a successfully completed operation. A red background reports failure. In Simple View, failure also causes the Build Status pane to automatically open, which displays detailed information about the failure.



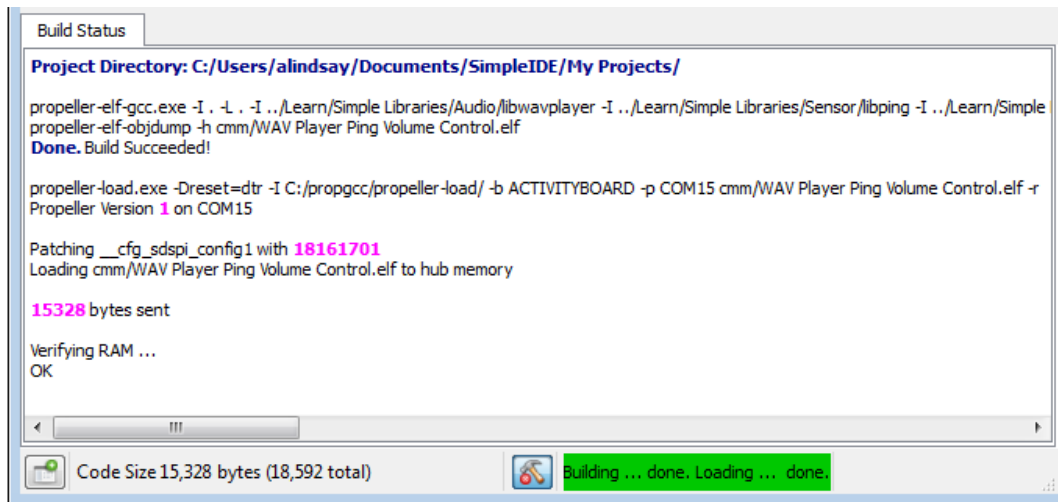
Download progress: Displays percent completed of code/data being transferred to Propeller chip and its peripheral memory.



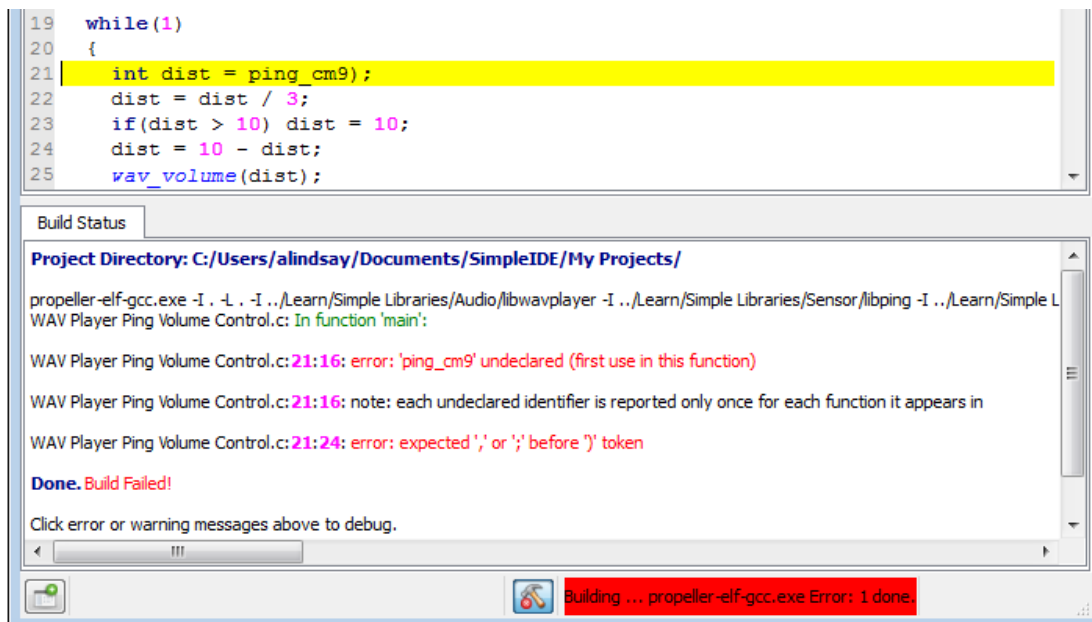
Build Status

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

In this example, the project directory is displayed first. After that, it shows information passed to the Propeller GCC compiler (propeller-elf-gcc.exe), and its report back (Build Succeeded!). It then shows the information passed to Propeller Load, the status of the program image transfer, bytes sent and verification that the program is in RAM in this case.

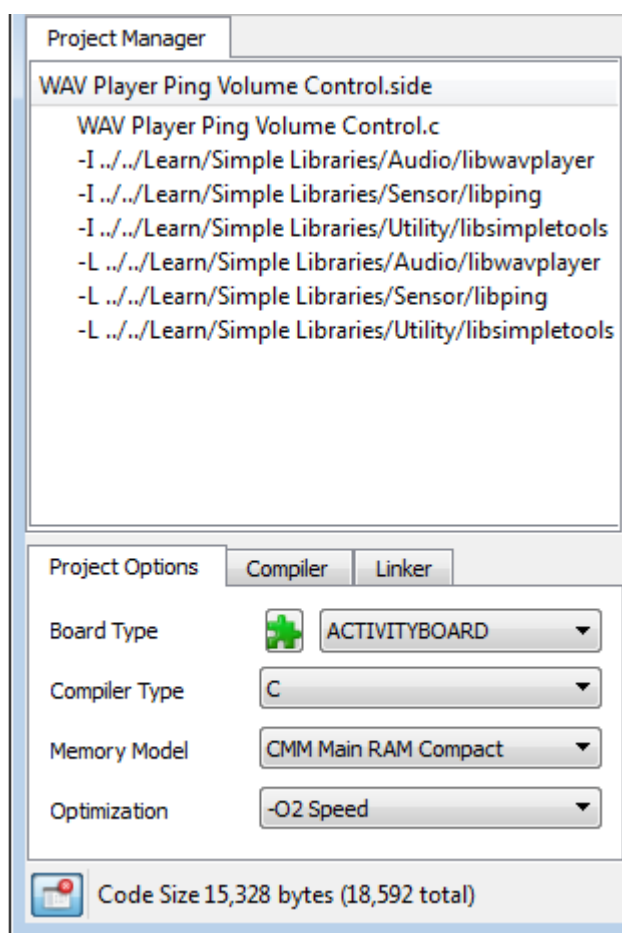


If there is a compiler error, the Build Status pane will report the line and character number where it detects each error, and the first detected error will be highlighted in the program. In this example, an opening parentheses was deleted on line 21 to cause the compiler error. It should read `int dist = ping_cm(9);`



Project Manager

Projects typically consist of several files. Those files must be listed in the Project Manager in order for the project to be built and run. The minimum required file in a project is the .c file with the main function (where code execution will start). In the example below, the project's name is: WAV Player Ping Volume Control.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.



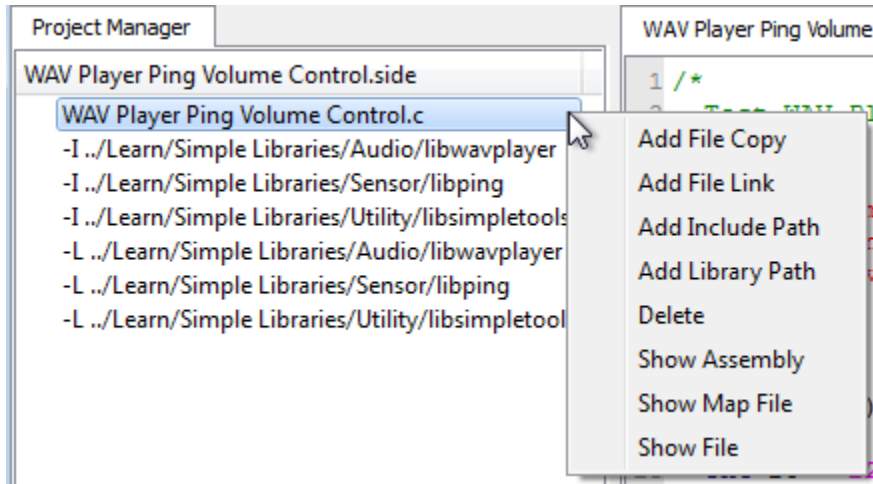
You can see how SimpleIDE uses the Project Manager to keep a list of any library, folder, and file resources the project utilizes that are not part of Propeller GCC. Libraries that are not part of propeller GCC, like the simpletools, ping and wavplayer libraries shown above have to be added to the project so that the compiler can find them. You will not see stdlib listed here because that is part of Propeller GCC, so a simple `#include <stdio.h>` in the program would be all that's required.

NOTES

- For C/C++ and Spin, the project manager defines what is built regardless of what is shown in the editor tabs.
- 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. The file will open in the editor tabs if the entry is a file. Entries starting with -I and -L specify paths, not files. Clicking the .side 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:** Add a copy of a file to the project. When adding a file to the project, it must already exist somewhere on the computer. A New file can be created in the editor and saved for example. If a file outside of the project directory is selected, the file will be copied to the directory. File names are added in alphabetical order except the main project file which must always be first.
- **Add File Link:** Add a link to an existing file to the project. When a link is added to a file, the file must already exist. The file will not be copied to the project directory. A link will have the short file name and -> full path name. A disadvantage of using links is not having all code in the same folder.
- **Add Include Path:** Add an include path for .h header files not in the project folder or tool-chain library folder. This adds an -I path to the project.
- **Add Library File:** Add a library file to the project. Add only ".a" library file(s) to the project. Files will not be copied to the folder.
- **Add Library Path:** Adds a path to a folder that contains one or more archived library binary images (.a files). This folder should contain subfolders with memory model names (cmm, lmm, xmmc, etc.), each with one or more .a files that have been compiled for that memory model. Assuming each folder contains libname.a, -lname should be added to the Linker tab's Other Linker Options field.
- **Delete:** To delete a file or link from the project, right-click on the item and choose delete. This action does not delete the file itself, it just removes it from the project settings. Do not delete the top file - it has a special meaning. The top file in the project must always have the main() function required by C/C++. If another top file is needed, either create it or open in in Project View, and then use Project -> Set Project to set it as the main file.
- **Show Assembly:** Right-click the file name and click Show Assembly to see the Propeller-GCC assembly with C source comments.
- **Show Map File:** Right-click the file name and click Show Map File to see the Propeller-GCC code map.
- **Show File:** This is the same as left clicking on a file name, and it'll jump to the tab with that file if it's already open, or open the file into a new tab if it's not.

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) is 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.

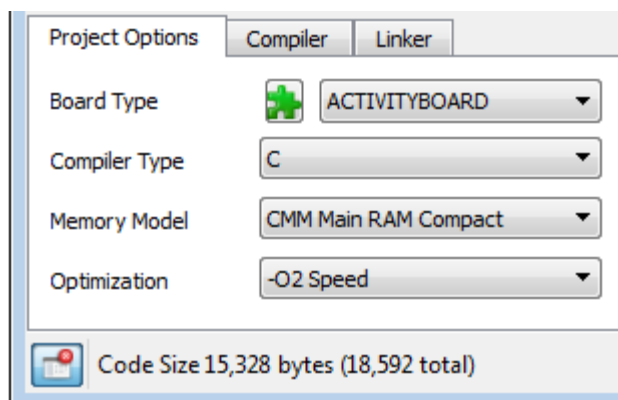
- Header files (*.h) are used to define data types and declare functions that may be in libraries.
- Header files are included in C source and do not need to be added to the project manager. However, it is a good idea to add a file link because it can then be conveniently opened into a tab.
- If the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

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 BSTC or other Spin programs used for making PASM COG code.

Project Options

The Project Options tab is for choosing typical Propeller-GCC project options. These options are automatically saved in the .side project file.



Board Type

This is the drop-down box on the right side of the green puzzle piece. Most commercially available Propeller boards are listed here in Project View, 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 or SDXMMC are special and when selected tell the IDE that certain functions have to be performed to load the program into external memory. 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 23 for more info.



Reload Board Types: If files have been added to or deleted from C:\propgcc\propeller-load*.cfg, this button will reload the names into the Board Type dropdown. If the C:\propgcc\propeller-load\boards.txt file has been modified, this button will update the truncated boards list in Simple View.

Compiler Type

Three compiler types are supported:

- C
- C++
- Spin

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

Spin is a custom object-based language (not related to C or C++) that Parallax developed for the Propeller microcontroller's multicore architecture. It is compiled by BSTC (Brad's Spin Tool Compiler), which is also bundled with SimpleIDE.

Memory Model

Memory model options allow you to select where code that gets executed, and data that gets accessed, are stored. There are six memory model options that utilize various combinations of the Propeller chip's 2 KB Cog Ram, 32 KB Main RAM (shared by all cogs) and various external memories including flash, SD, and even the EEPROMs (64 or 128 KB recommended) built into many Propeller boards.

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

- **LMM Main Ram:** Large Memory Model program image stored in Main RAM with machine codes fetched and executed by one or more cogs. Variable data is also stored in Main Ram.
- **CMM Main Ram:** Compact Memory Model size optimized program image stored in Main RAM. Like LMM, machine codes are fetched and executed by one or more cogs. Variable data is also stored in Main Ram. This is recommended for most applications that fit in the Propeller's 32 KB Main RAM. There is very little difference in performance between LMM and CMM, and the memory savings is significant.
- **COG Cog RAM:** Both program image and data reside in Cog RAM. Only small amounts of function local variable data can reside in cog memory. COG programs are very limited. VGA-Pong and VGA-driver demo code are some surprising examples that will run in COG using Main RAM for buffering. PASM not required.

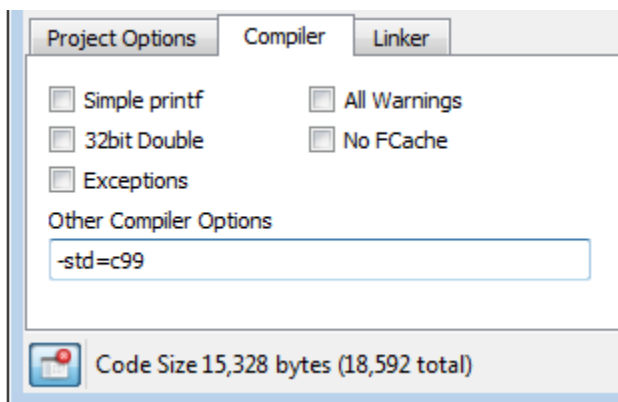
- **XMMC External Flash Code Main RAM Data:** Extended Memory Model Code program image stored on flash memory, SD card, or EEPROM. Data such as variables are placed in Main RAM.
XMMC is the best performing external memory model. It is faster than Spin programs in many cases when using SPI Flash. Code can be forced into Main RAM for best performance of time critical tasks. This is a solution for the XMM determinism problem. XMM code is not deterministic “a-priori”. XMM code relies on a cache for performance and fetching cache lines can get in the way from one compile to the next depending on the code being used.
- **XMM-SINGLE External RAM:** Extended Memory Model Single stores both program image and data on an external RAM. Data can also be forced into Main RAM.
- **XMM-SPLIT:** External Flash Code External RAM Data – stores the program image on an external flash, SD, or EEPROM memory and variable data on an external RAM.

Note: XMM is compatible with libraries that launch COGC or PASM code into other cogs. However, it is not currently compatible with code that launches LMM or CMM code into other cogs.

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. Another example, the Graphics demo can run on CMM or LMM, but it can also run on XMMC with board type EEPROM selected (A 64 KB or greater EEPROM must be used; two 32 KB EEPROM ICs will not work.), Approximately 6 KB of EEPROM code is used for program resources with EEPROM XMMC.

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 for speed optimizations.

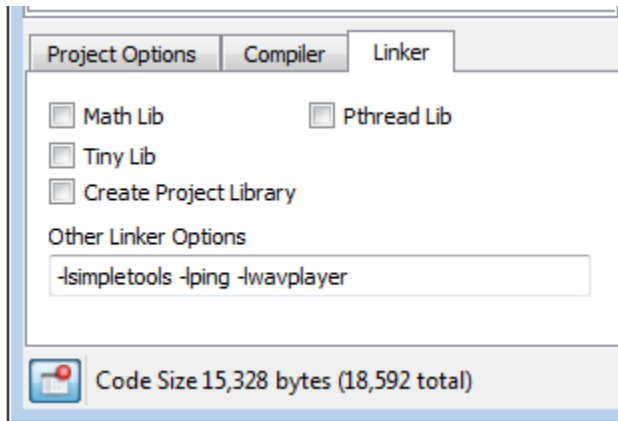
Compiler Options



- **Simple Printf:** Deprecated. See Linker tab's Tiny lib option.
- **32bit Doubles:** Use 32-bit doubles for floating point double variables. The default is 64-bit doubles, and may be too big for most LMM programs.
- **All Warnings:** Tells compiler to generate all possible warnings on issues in code that can cause trouble.
- **No Fcache:** This tells compiler to not use Fast Cache (fcache). Fcache generally improves performance but it can be disabled.

- **Exceptions:** This should be enabled for C++ programs that use try/catch exceptions. Using exceptions may cause code size to increase.
- **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 Program Manager's Add Include Path for using prebuilt libraries. One may need -I <path-to-library-headers> for using prebuilt libraries.

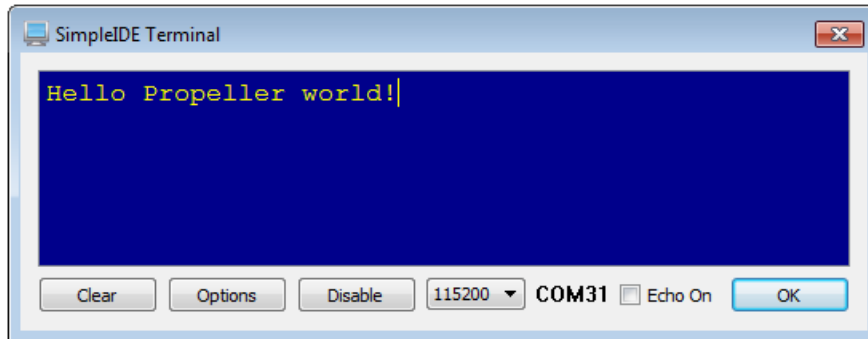
Linker Options



- **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:** Significantly reduces code size for programs that need `printf`, but not file I/O or math library support. For example, the Welcome Application's simple hello message drops from 8.21 to 3.16 KB.
- **Pthread Lib:** This option must be checked if using Pthreads for running multiple threaded programs in one cog or many cogs. The number of threads available is limited by memory. XMM/XMMC programs will run all threads on the main program cog. LMM programs can run M threads on N cogs. For Pthreads N cogs is limited to 8 for LMM programs and 1 for XMM programs. M threads is limited only by memory available. This is an advanced feature; see 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 is that has subfolders with memory model initials like cmm, lmm, xmmc, etc. Each will contain 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 results of `printf`, `putc` and other terminal display functions. The window can also be clicked to enter text that can be received by the Propeller with functions like `scanf` and `getc`. Data from this window can also be shaded, copied, and pasted into other documents.

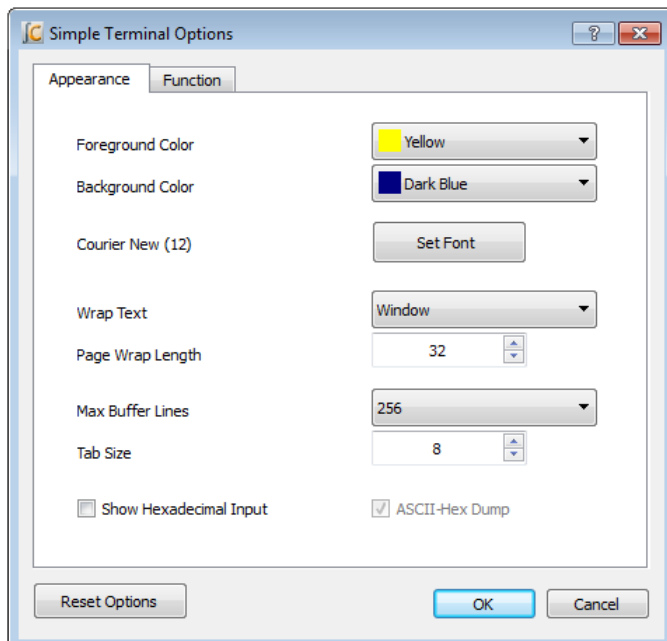


Features

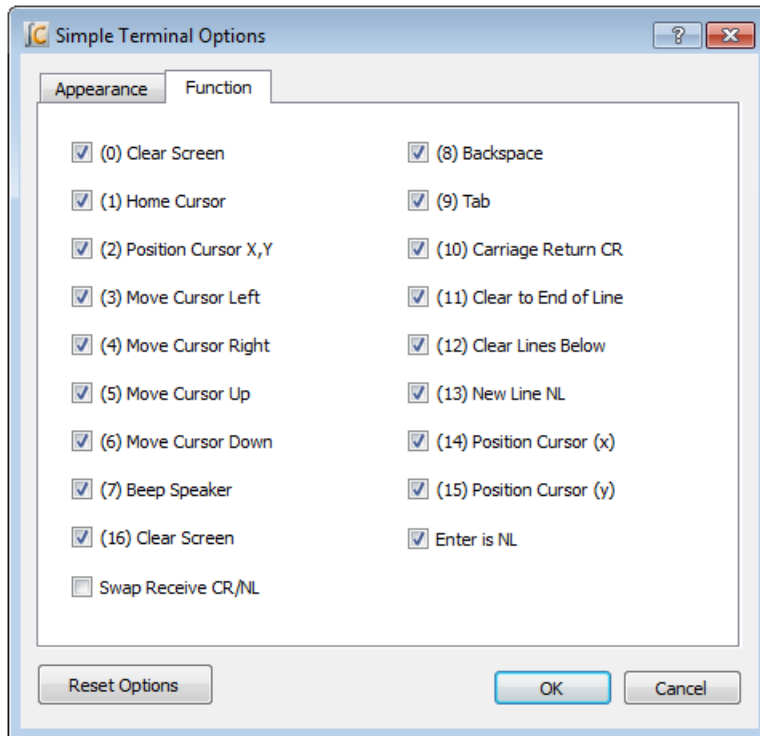
- A Clear button for clearing the terminal of any previously printed text
- An Options button for setting Appearance and Function
- Disable button to stop displaying incoming messages
- A baud rate selection dropdown
- Com Port that is connected
- Echo On checkbox
- OK button to close the window.

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

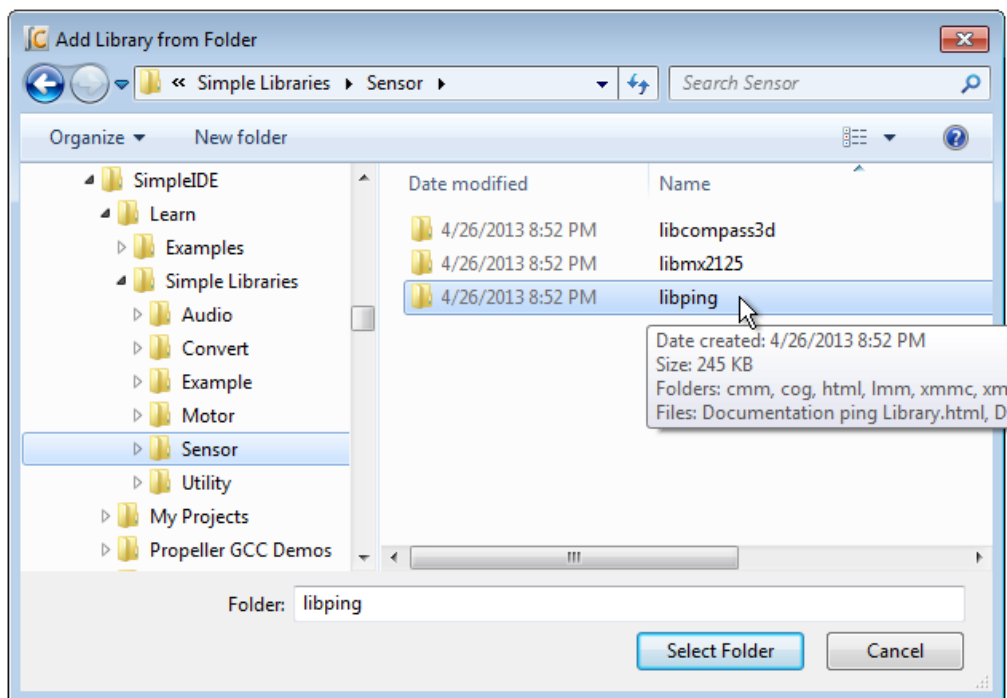
Simple Libraries are designed to be easy to add to a project with SimpleIDE. A number of Simple Libraries, including simpletools, are included in Documents > SimpleIDE > Learn > Simple Libraries.

How to add a Simple Library to a Project

1. Click the Add Simple Library button.

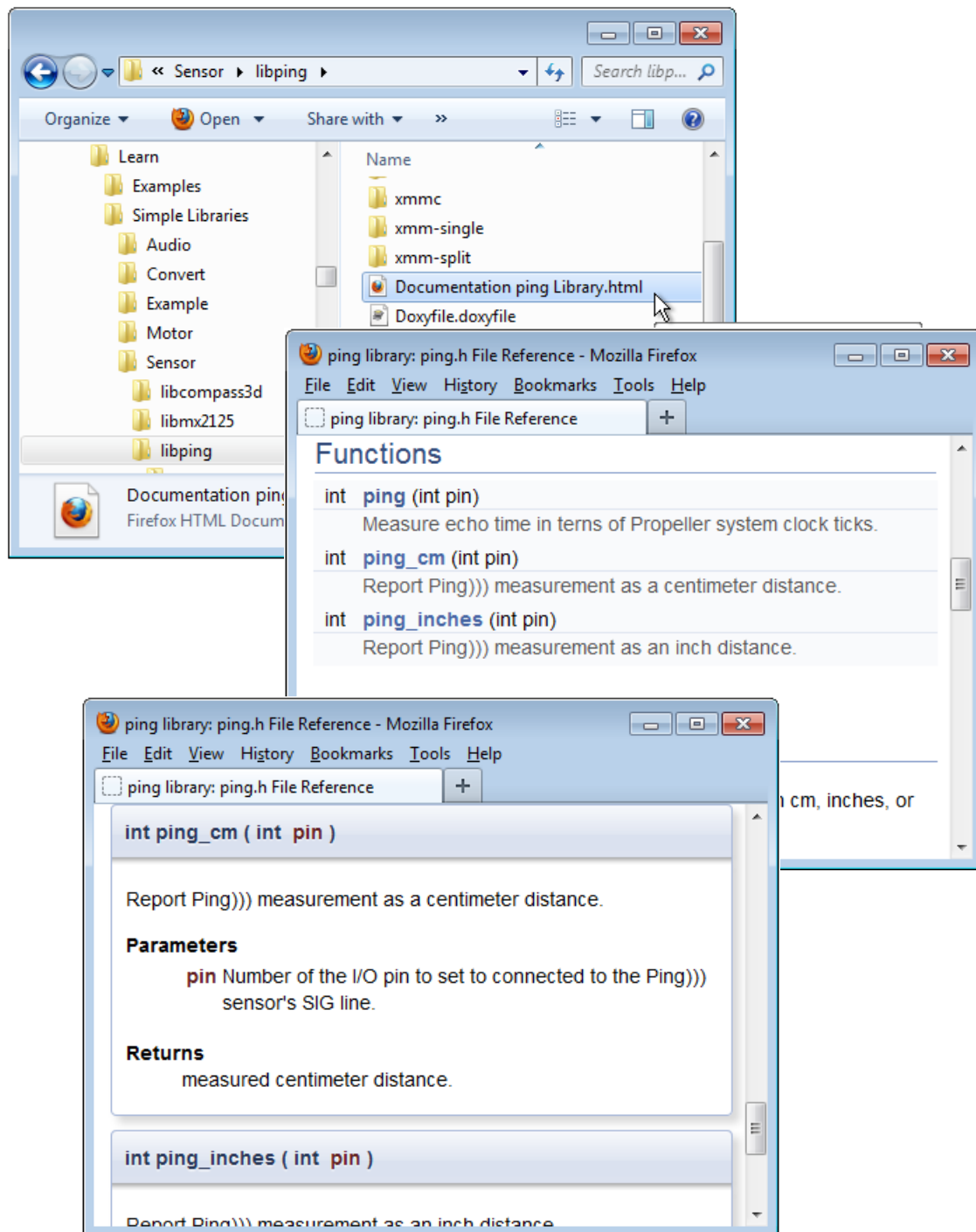


2. Browse to the library you want to add, select it, and click the Select Folder button.



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

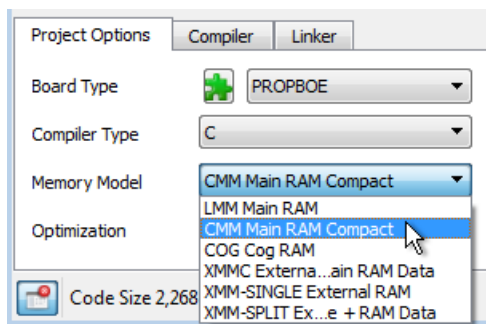
Simple Library Documentation: Use your operating system's file browser (Windows Explorer, Mac Finder) to look inside the Simple Library folder and open the Documentation...Library.html file. Among other things, this documentation lists the functions the library puts at your disposal, describes what they are for, the parameters they expect and what they return.



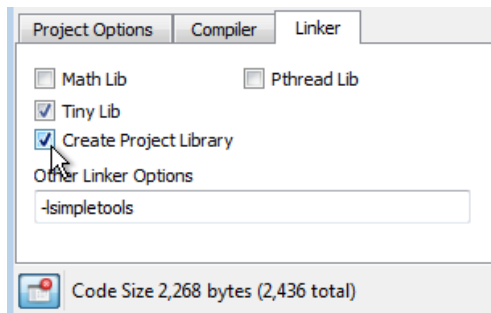
How to Create a Simple Library

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

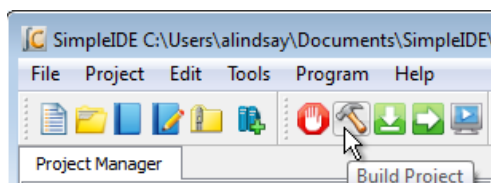
1. Create a project named libname within a folder named libname.
2. Use Project -> Add Tab to add name.h to the libname folder.
3. Add as many .c files as needed, and try to make the names descriptive of the functions they contain. Also try to make each .c file contain as few functions as possible because the archived version will optimize out uncalled functions if you do.
4. To create the memory model subfolders with the .a files, click Show Project Manager (if you're in Simple View), and select the memory model for the .a file you want to create in the Project Options tab.



5. Place a checkmark in the Linker tab's Create Project checkbox.

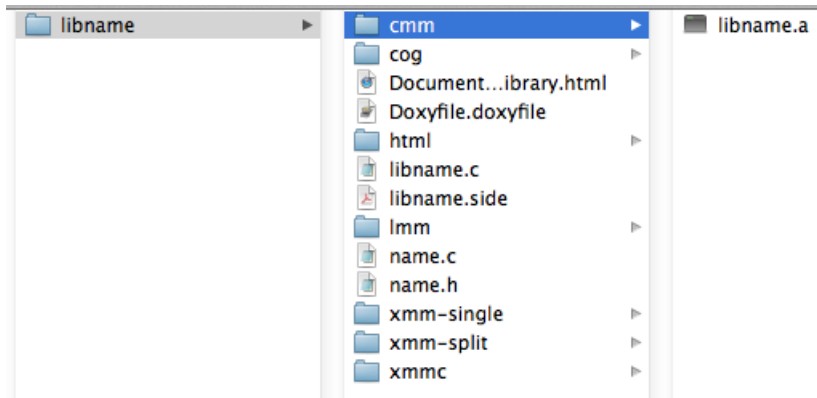


6. Click the Build Project button.



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, xmmc, or other memory model subfolders. Check subfolders in ... \SimpleIDE\Learn\Simple Libraries for examples. The Documentation Name Library.html file and html folder are discussed in Recommended Features.

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. After the .a file is created, it will cause program images to only expand by the size of the functions that are called (along with any they depend on). In contrast, if a single, long, wandering .c file with every function in the library is used, one function call will increase the program image size by the code for all functions within the library.

Doxygen (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 the drop down box. There are many board types and variations. Board type .cfg files contain information for the loader to use for 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 modifying the program. More details can be found in the propeller-load document.

Special Clock Boards

If the Propeller board does not use 80 MHz or has an odd crystal configuration, a board type can be created to set the clock frequency (typically crystal frequency times PLLx mode). Board configuration files can be set in the project folder, but it is best to put them in the installation propeller-load folder.

Assuming C:\propgcc is the installation directory, define a custom board as follows:

1. Copy the C:\propgcc\propeller-load\hub.cfg to c:\propgcc\propeller-load\custom.cfg (or choose some custom 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, open boards.txt (from same directory), 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 on reload.
4. Click the puzzle piece  in the Project Option control to reload board types.
5. Choose CUSTOM from the board type drop down box.

Note: RCFAST and RCSLOW board types are available in the board types, but these should never be picked for programs that communicate through SimpleIDE Terminal.

Basic Board Types

Basic board types can work on many boards that have only a Propeller, crystal, and EEPROM. A crystal is not necessary for RCSLOW and RCFAST board types. 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 those whenever the board type is known in advance. GENERIC settings include 5 MHz crystal with PLL set to 16x for 80 MHz system clock, 32 KB or larger EEPROM connected to P29 (SDA) and P28 (SCL), and 115 kbps communication with terminal via P30 (host computer Rx) and P31 (host computer Tx).
- **RCFAST:** This board type makes the Propeller chip rely on its internal 12 MHz oscillator. Although it can boot into any propeller board, it is designed for applications that do not require

the clock precision of an external oscillator. This mode is not recommended 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.
- **HUB:** The HUB board type specifies an 80 MHz system clock and an external 5MHz 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 HUB board type and a good 5 MHz crystal. Other board types related to HUB are ASC, ASC, QUICKSTART, PEKIT, PROPBOE, ACTIVITYBOARD, and others.
- **SPINSTAMP:** The SpinStamp board type is like the HUB board type except that it relies on a 10MHz 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.

EEPROM Board Types

The EEPROM board type is like the HUB board type except that it has a cache driver defined for running XMMC memory model programs. With the EEPROM or similar board types, a program loaded into the EEPROM can be fetched and run with XMMC.

This mode is usable with single 32 KB EEPROM, 64 KB EEPROM, 128KB linear address space EEPROMs, and 256 KB EEPROMs from ST.

Any set of 64 KB+ EEPROM can be configured to add more code space to the program. Using two separate 32 KB EEPROMs will not work because address 0x8000 to 0xFFFF maps to address 0x0000 to 0x7FFF. All devices should be the same type. MCP29FC1025 devices do not have a linear memory addressing and will not work.

The XMMC model is the only extended (XMM*) memory model that will work with EEPROM board types. Other board types related to EEPROM that run XMMC programs are ACTIVITYBOARD, PROPBOE, QUICKSTART, and ASC+.

External Flash Board Types

External Flash board types like C3F and SSF will run XMMC memory model programs from Flash memory.

- **C3F:** C3F XMMC programs are stored in the on board 1MB SPI Flash. C3F is a variant of the C3 board type that uses all of cache for program code and is thus faster than C3. (The C3 type splits cache between Flash and SRAM storage and is less efficient).
- **SSF:** SSF is the SpinSocket-Flash board type that uses 2 Winbond W25Q* QuadSPI parts for storing and running XMMC memory model code in. This is a fast practical external memory solution because of low cost, low pin count, high density, and high relative XMM performance.

External RAM Board Types

External RAM boards have external SRAM, SPI-SRAM, or SDRAM. Boards having only SRAM will only boot stand-alone if the code can be loaded from SD card or some other non-volatile storage. SRAM only boards can be loaded by the PC and SimpleIDE using the serial loader protocol for testing.

- **C3:** The C3 type allows using the XMM (or XMM-SPLIT) memory model to store program code in SPI Flash and data in a device like external SPI SRAM. It is possible for a cache driver to be

written for other boards that will use the XMM memory model. The C3F board type will only use C3 Flash and only works with XMMC memory model programs.

- **DRACBLADE:** This board has SRAM and SD card. It can be loaded by the IDE for testing, but must be programmed using SDLOAD for stand-alone boot.
- **SDRAM:** This board has SDRAM and SD card. It can be loaded by the IDE for testing, but must be programmed using SDLOAD for stand-alone boot.

SDLOAD Board Types

SDLOAD board types are typically RAM board types that do not have an on board Flash. With the SDLOAD board type, the program to be run is sent to the SD card. The Load RAM & Run, Load EEPROM & Run, or Run with Terminal buttons will cause the AUTORUN.PEX output to be downloaded to SD card and then booted to RAM and run. Once the Load EEPROM & Run button has been used, the AUTORUN.PEX file can be replaced using the Send File to SD Card command/button or by connecting the SD card to your computer and replacing it with file management tools like Windows Explorer or Mac Finder.

SDXMMC Board Types

SDXMMC board types are used to download XMMC memory model AUTORUN.PEX programs to SDCard and run them using the Load RAM & Run, Load EEPROM & Run, or Run with Terminal. Any board that has an SD card can use SDXMMC; however, code execution may be very slow.

SimpleIDE SDLOAD and SDXMMC Attributes

How does a board type include SDXMMC or SDLOAD attributes? This is added to the .cfg file for a board. If a board has an SD Card, the SDXMMC option can be added to the .cfg file with the line "# IDE:SDXMMC". If a board has an SD Card and a supported RAM cache driver, SDLOAD can be added to the .cfg file as "# IDE:SDLOAD". Some examples follow.

Below is an example of the ppushb.cfg file. Note that it has IDE:SDXMMC in a comment as described above. Other interesting items are cache-driver and sd-driver. The cache driver in this case is the eeprom_cache.dat driver. Today, the SDXMMC cache driver is part of the sd_driver.dat.

To use SDXMMC for this example, the board type PPUSB-SDXMMC should be selected. To use the EEPROM cache for XMMC, the board type PPUSB should be selected.

```
# ppushb
# IDE:SDXMMC
  clkfreq: 80000000
  clkmode: XTAL1+PLL16X
  baudrate: 115200
  rxpin: 31
  txpin: 30
  cache-driver: eeprom_cache.dat
  cache-size: 8K
  cache-param1: 0
  cache-param2: 0
  eeprom-first: TRUE
  sd-driver: sd_driver.dat
  sdspi-do: 0
  sdspi-clk: 1
  sdspi-di: 2
  sdspi-cs: 3
```

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/>). 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(CLKFREQ/2+CNT);
            OUTA |= (1 << _cfg_ledpin);
            waitcnt(CLKFREQ/2+CNT);
        }
    }
    while(1);
    return 0;
}
```


Future Improvements

Opportunities for improving SimpleIDE listed below. Email ideas to gccbeta@parallax.com.

1. Auto-install the FTDI USB driver for communication with Parallax boards.
2. Propeller auto-detect.
3. Automatically add simple libraries to projects based on #include directives in both main file and included libraries.
4. Add a built-in browser for allowing F1 Context sensitive help and opening the SimpleIDE User Guide and library documentation.
5. Code autocomplete - Add library function and object "dot" lookup to make programming easier.
6. Debugging tool.

Support

Report problems with SimpleIDE or Propeller-GCC via email to gccbeta@parallax.com or by adding issues to <http://code.google.com/p/propside/issues/list>

Revision History

There have been many untracked enhancements and bug fixes during the life of this document. In the future, significant bug fixes may be tracked here. For complete details, look at propside.googlecode.com for complete software revision history and source files.

- 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.