

Simple IDE User Guide

Copyright © 2012 Parallax, Inc.
Written by John Steven Denson

The Simple IDE is designed for programming the Parallax Propeller in C/C++.

Table of Contents

SIMPLE IDE FEATURES	3
SCOPE	4
INSTALLING SIMPLEIDE	5
WINDOWS INSTALLER	5
MAC OSX	5
LINUX PACKAGING	5
STARTING THE IDE THE FIRST TIME	6
START-UP "ABOUT" SPLASH	6
SIMPLEIDE PROPERTIES	6
FOLDERS	7
GENERAL SETTINGS	8
SYNTAX HIGHLIGHTING	9
IDE CONTROLS OVERVIEW	10
FILE MENU	10
PROJECT MENU	10
EDIT MENU	10
FIND AND REPLACE	11
TOOLS MENU	11
PROGRAM MENU	12
SERIAL PORT CONTROL	12
HELP MENU	13
HELLO WORLD DEMO	14
PREPARING THE SERIAL PORT	14
SELECTING BOARD TYPE AND SERIAL PORT	15
RUNNING THE HELLO DEMO	16
CREATING A NEW PROJECT	17
OPTION 1: NEW PROJECT	17
OPTION 2: CLOSE ALL, USE AN EXISTING MAIN FILE	17
OPTION 3: CLOSE ALL, START NEW	18
PROJECT MANAGER	19
PROJECT FILE TYPES	20
<i>Source Files</i>	20

<i>Include Files</i>	20
<i>Object Files</i>	20
PROJECT FILE LIST	21
PROJECT OPTIONS	23
BOARD TYPES	26
SPECIAL CLOCK BOARDS.....	26
BASIC BOARD TYPES.....	26
EEPROM BOARD TYPES.....	27
EXTERNAL FLASH BOARD TYPES	27
EXTERNAL RAM BOARD TYPES.....	27
SDLOAD BOARD TYPES	28
SDXMMC BOARD TYPES	28
SIMPLEIDE SDLOAD AND SDXMMC ATTRIBUTES	29
CONFIGURATION FILES.....	30
CONFIGURATION VARIABLE PATCHING	30
SOURCE BROWSING	31
FUTURE IMPROVEMENTS	32
MISCELLANEOUS NOTES	32
SUPPORT	32

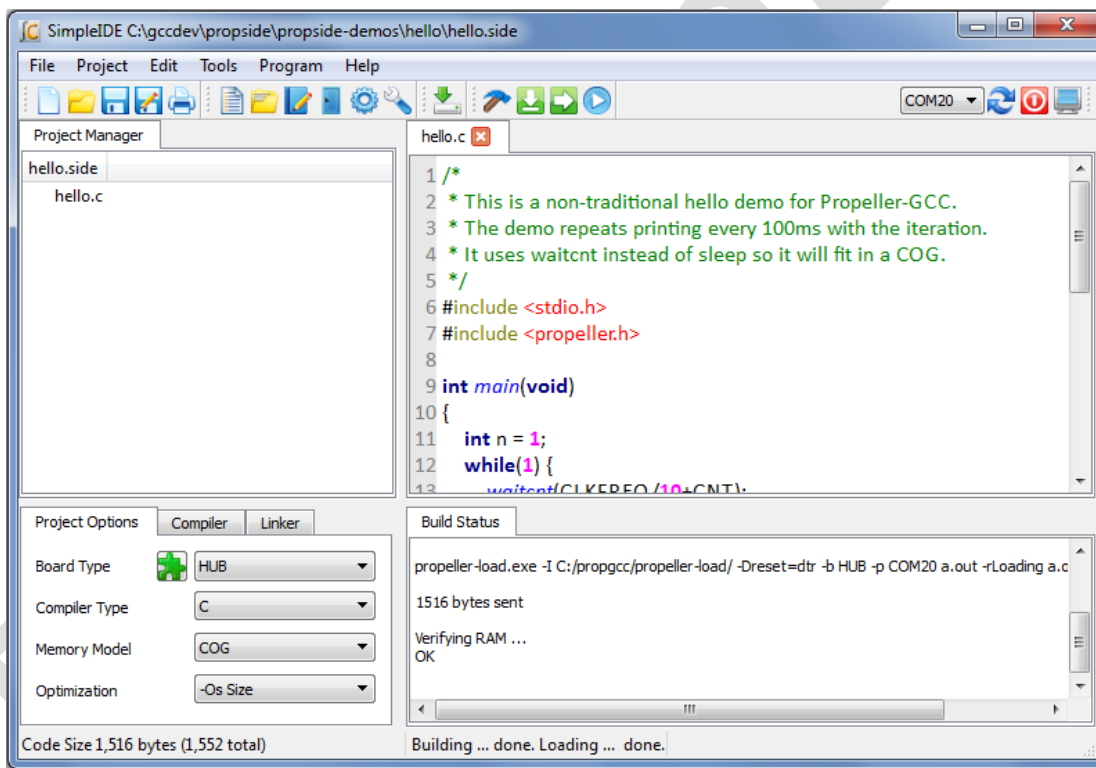
Document Revisions:

Preliminary May 26, 2012 untracked.

Preliminary Beta June 16, 2012 untracked.

Simple IDE Features

- Menu Bar with File, Project, Edit, Tools, Debug, and Help
- Tool Bar with most Menu Bar operations
- Allows Cloning and Saving current projects
- Project manager menu and area for project file settings
- Compiler area for setting build attributes
- Tabbed text editor with optional highlighting
- User selectable font size and family
- Source Browser finds declarations
- Project build status pane
- Build Status shows build progress
- Status bar shows compile size, brief messages, and progress bar
- Board type and COM/serial port selector
- Available for Linux, Mac OSX, and Windows
- Multiple language interface and code capable



Scope

This guide is not comprehensive. It describes the basic features of SimpleIDE by describing installation, first startup, first program, over-view of IDE controls, details of using project management features, and board configurations. This guide has some C program examples, but it does not attempt to teach programming.

SimpleIDE is designed primarily as a platform for anyone interested in writing programs and managing program builds in a simple way for the Propeller-GCC compiler tool chain. It may be expanded in the future for other compilers.

SimpleIDE does not use the make utility at this time for building programs and it is not discussed here. However users can use make with the Propeller-GCC package that is included with SimpleIDE.

Uses of personal pronouns in this document are accidental.

Preliminary

Installing SimpleIDE

SimpleIDE is distributed as a Windows InnoIDE install package, a Mac OSX zip (.zip), Linux i686, Debian x86_64, Fedora x86_64, and is available as open source to compile with Qt 4.8 and GCC.

Windows Installer

The windows installer is about 76MB and contains the Propeller-GCC tool-chain, Simple IDE, and Propeller Demo programs installed in the user's selected workspace. The installer will ask the user for directories and other info for setup. The installer makes the package usable in one simple process.

A properties dialog will open the first time, and the Compiler tab should always have the correct information for windows. Just click OK to get started.

Mac OSX

The SimpleIDE.zip is about 60MB and contains the Propeller-GCC tool-chain, Simple IDE, and Propeller Demo programs in the file. Use finder to unpack the contents of the .zip to the user folder. Once the .zip is unpacked, click on the app icon to start SimpleIDE.

A properties dialog may open the first time, and the Compiler tab should always have the correct information except for the user workspace.

Linux Packaging

Linux packages are primitive at the moment and will be changed to use the typical packager tools for the platform. The current package includes a setup script and supporting shared libraries. If items are missing from the system required by SimpleIDE the ldd can be used tool to find out what is needed, I.E. `$ ldd SimpleIDE`

This Linux SimpleIDE package contains only the SimpleIDE. It does not contain the Propeller-GCC tool-chain. Debian packages are known to work on Ubuntu.

In the instruction below SimpleIDE-packagename.bz2 refers to the package such as SimpleIDE-0-6-2-i686-linux.bz2. SimpleIDE-version refers to the SimpleIDE version such as SimpleIDE-0-6-2

1. Download and install Propeller-GCC to /opt/parallax
2. Unpack SimpleIDE with `$ tar -xjf SimpleIDE-packagename.bz2`
3. Change directory `$ cd SimpleIDE-version.`
4. Setup with `$./setup.sh`
5. Run with `$./simpleide`
6. In SimpleIDE Compiler Properties:
 - a. Choose the compiler from /opt/parallax/bin
 - b. Choose the loader path as /opt/parallax/propellerl-load
 - c. Choose a workspace for new projects.
7. Open the hello demo from SimpleIDE-version/demos/hello/hello.side

Starting the IDE the first time

Start-up "About" Splash

At first startup the SimpleIDE "About" window appears. This shows the current version which can 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, and an OK button.

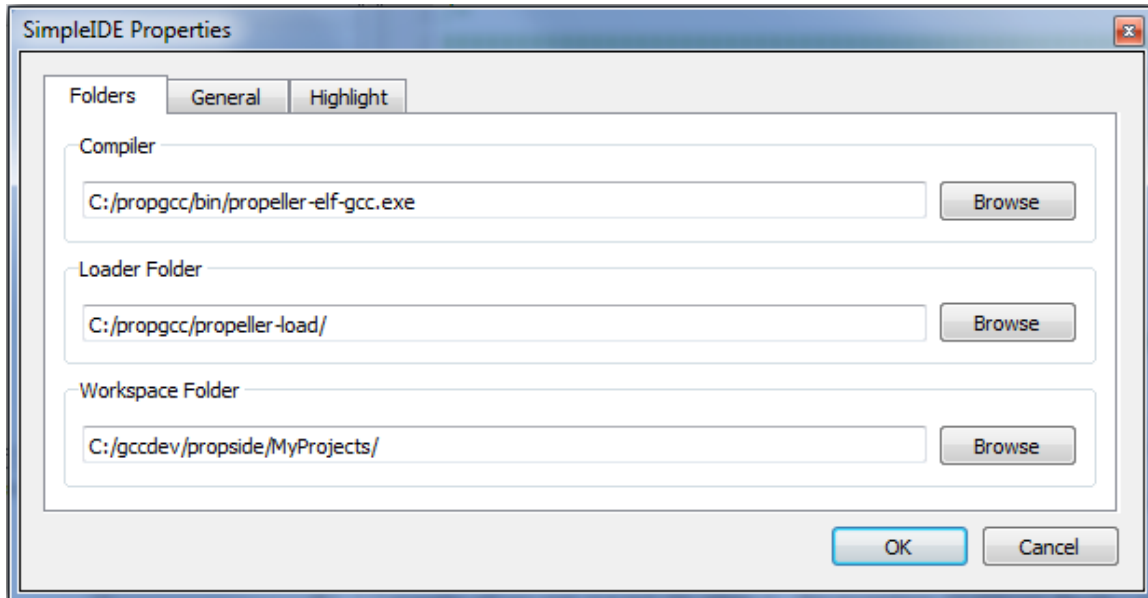


If the "Show this window at startup" is checked, the window will appear on every startup. Clear the check box to disable showing this at every startup.

SimpleIDE Properties

At first startup a dialog window titled SimpleIDE Properties may appear. It allows setting key Folders, General properties, and Highlights.

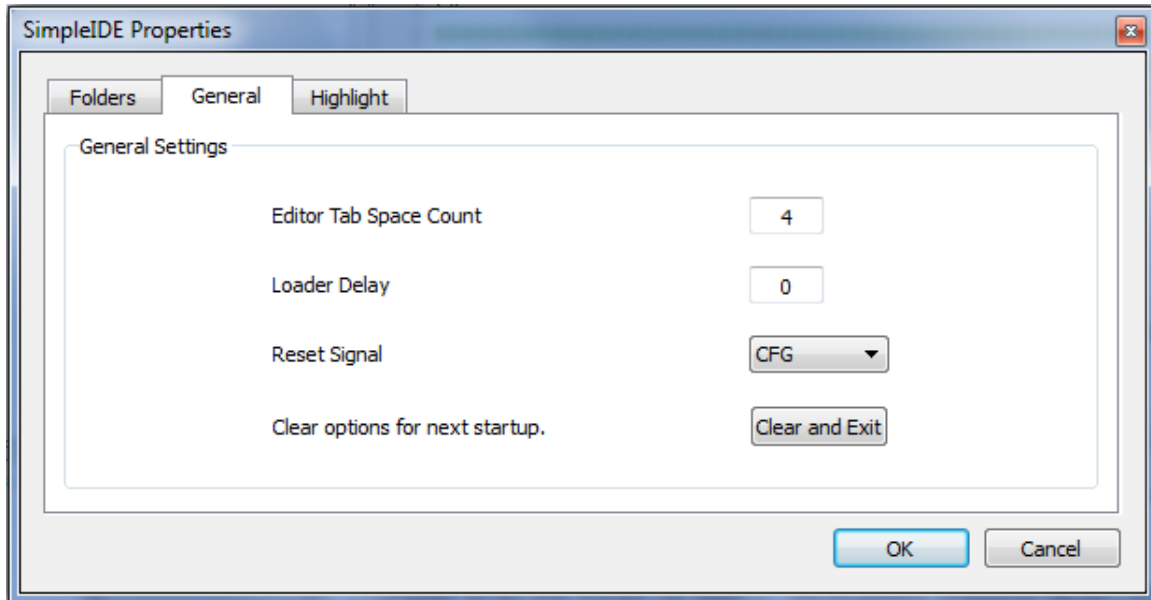
Folders



In most cases, the Folders tab will already have the fields properly set. The backslash '\' folder separators used in windows are replaced by '/' in the IDE. Please take note of the fields and click OK when ready. The Compiler and Loader Folder must be set correctly before moving on to the next step. Please choose a workspace if one is not already set.

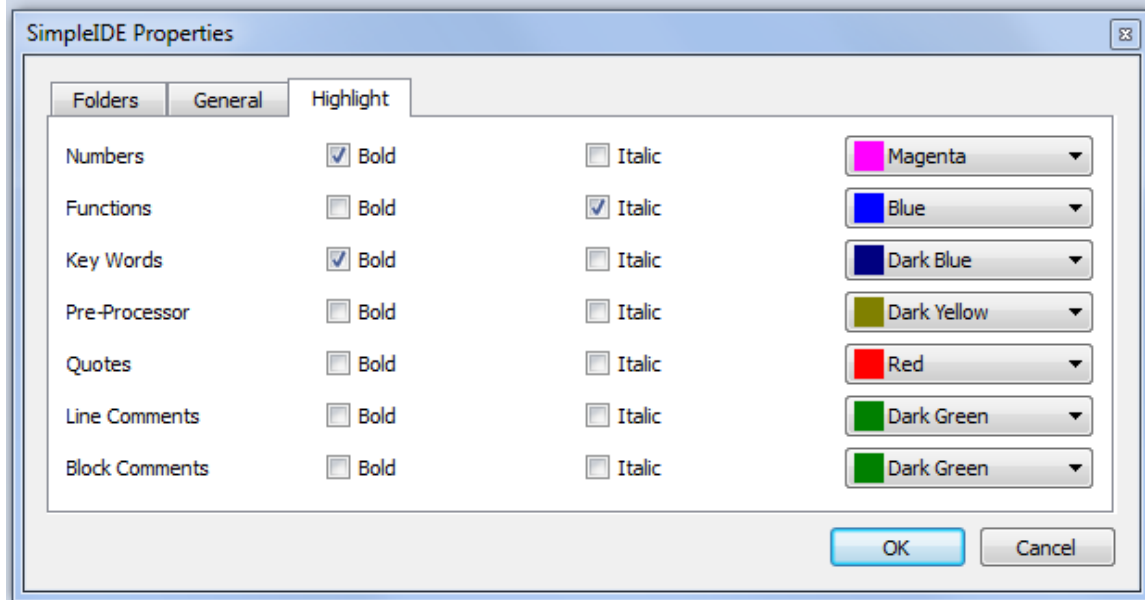
All of these properties may be changed when the IDE is running to allow using another compiler directory by clicking the tool-bar wrench. If the window reopens after clicking OK, it means that the critical "Compiler" or "Loader Folder" information is not correctly set. In most cases this will not be a problem, but if it is the Browse buttons can be used to find the propeller-elf-gcc compiler program and propeller-load folder.

General Settings



Some general property details may need to be changed for different boards. For example, some USB serial devices do not have DTR for controlling Propeller reset and should use RTS instead. The Reset Signal can be changed from DTR to RTS if necessary. The CFG option chooses the reset signal type specified in the board configuration file.

Syntax Highlighting



The Highlight properties tab can be used to change editor syntax colors. At this time only a select set of system colors is available.








Open the SimpleIDE Properties window any time by clicking the toolbar wrench button. 

The SimpleIDE Properties dialog window controls are the same regardless of user's operating system.







IDE Controls Overview

Many features of the IDE will be familiar to new users; some features need explanation. Reviewing this section first may help understanding other content.






File Menu

-  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

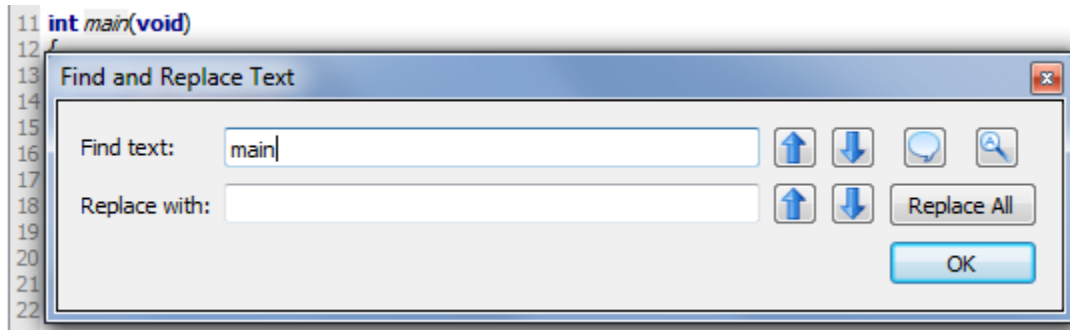
-  New Project: Opens a dialog for creating a new project in "folder/name" selected by user. Project name will be added to the folder by the program from the user's project name. If the folder/name does not exist, it is created.
-  Open Project: Opens an existing project file with extension .side
-  Save As Project: Save the current project with a new name to a folder. This will save all project settings and relative paths, then open the project.
- Clone Project: Works the same as Save As Project except the source project can be selected before saving.
-  Save and Close Project: projects are always saved before they are closed.
-  Set Project. The function of the set project button (F4 or Project->Set Project) is used to make a project using the currently visible user program.
- Previous project names: Lists the last 5 opened projects.
-  Properties. This will open the SimpleIDE properties menu which was discussed in the first time use section.

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 with find and replace tools that allows for optional whole word and case sensitive find and replace.
-  Redo: Undoes the last undo.

-  Undo: Reverses the last edit.







Find and Replace







The Find and Replace dialog window allows searching and replacing text in the editor. Find does not allow pattern matching yet.

- 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 previous or next buttons.
- Replace with: when a word is entered in this field, it will be used to replace the find word when the replace previous or next button is clicked. The Replace All button will replace all instances of the find word with the replace word in the current editor.
-  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.

Tools Menu

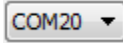



-  Send File to Target SD Card: If the board-type has SDLOAD or SDXMMC in the name, this button can be used to send any file to the SD Card. Typically clicking this button will also compile the project to an AUTORUN.PEX image that can be used when Propeller boots up.
-  Go Back: When not greyed, pressing this button will show the last browse location.
-  Browse Declaration: Allows zooming to a user defined function or global variable declaration. Library functions such as printf that are not in the project file-list can't be browsed.
-  Font: Allows selecting an editor font.
-  Bigger Font: Increases the editor font by 20%.
-  Smaller Font: Decreases the editor font by 20%.
- Next Tab: Lets user scroll to the next tab in the editors.

Program Menu

-  Run Console (F8): Build, load, and run program on Propeller RAM (or external memory for XMM). It opens a serial port console terminal window.
-  Build (F9): Build program only.
-  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.
-  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.




Serial Port Control



-  Serial Port: This is the drop-down box on the left of the rescan button. It lets the user choose the port to use where the board is attached. Any port can be selected. If the mouse hovers over the control, the "hover help" may tell give more information such as whether the port is a USB serial port or Bluetooth.
-  Rescan: Click the rescan button every time port hardware connections change.
-  Port Reset: This red "power switch" button allows resetting the board.
-  Serial Port Console: 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.

Help Menu

The help menu gives users access to documents for the IDE and Propeller GCC Library.

-  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.
-  Help: Shows link to the Propeller-GCC site.
www.parallax.com/propellergcc

Preliminary

Hello World Demo

Traditionally, a "hello" demo is the first program run with a C compiler tool set. A hello program is included in the demo package. It will be the first program compiled, loaded, and run. Later we will use a blinker program to show starting a new project.

SimpleIDE uses a separate loader and terminal program. The hello program run with SimpleIDE demonstrates a weakness in this arrangement. That is, the program must delay printing otherwise the output can be lost in transition.

A minimum hello program used in SimpleIDE will look like this.

```
#include <stdio.h>
#include <propeller.h>

void main(void)
{
    /* one second delay before printf */
    waitcnt(CLKREQ+CNT);
    /* traditional hello message */
    printf("hello, world\n");
}
```

The demo code included in the package runs a loop and prints the loop iteration.

Before running anything on Propeller, there are a few simple things to do. That is, preparing the serial port and setting a board type.

Preparing the Serial Port

To use SimpleIDE with Propeller on a computer having only USB ports it may be necessary to install an FTDI USB/VCP driver.

Windows FTDI USB/VCP Installation

Please follow directions described here

<http://www.parallax.com/Accessories/USBDrivers/tabid/530/Default.aspx>

If that page is not found, please visit www.parallax.com page to find the USB drivers page.

Mac OSX FTDI USB/VCP Installation

FTDI drivers should be preinstalled for OSX. Simple IDE packages do not support other OS versions.

Linux OS FTDI USB/VCP Installation

Linux users can install FTDI USB/VCP drivers using their package manager.

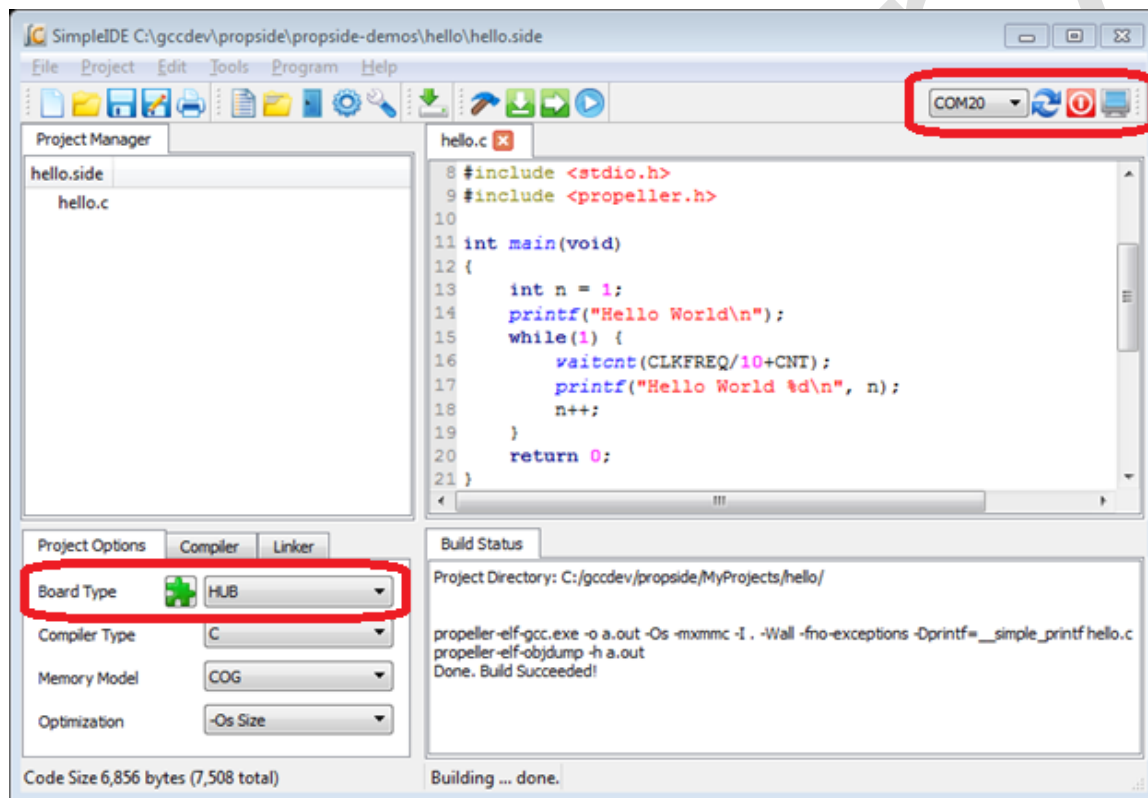
In other cases, if there are problems with the drivers, one can visit the FTDI driver web page at <http://www.ftdichip.com/Drivers/VCP.htm> and follow their instructions.

Selecting Board Type and Serial Port

To run the Hello demo, a Propeller board must be connected, a suitable board type selected, and the right COM port selected. The COM port will only appear if the Propeller board is connected.

The picture below shows the location of the board type and serial port selectors. In this case the board is set to HUB, and serial port set to COM20.


The board type is part of the project manager side bar because it is saved per project. In this case board type HUB is used for this "hello.side" project. The serial port is not saved in the project because it is more of a general setting.



The Hello demo should be run with a simple board type like HUB like shown above.

The ASC board type is suitable for most 5MHz Propeller board tests. Browse the board types and see what else is available. HUB is the most general board type for 5MHz crystal hardware. Some boards have a different clock types. The HYDRA and SPINSTAMP boards are good examples that use a 10MHz crystal.



If SimpleIDE is started without a serial port device connected, the COM port (shown as COM20) will be blank. In that case, connect a USB port Propeller board to the computer and click the rescan button. Demos can't be run without hardware.

The rescan button looks like this: 

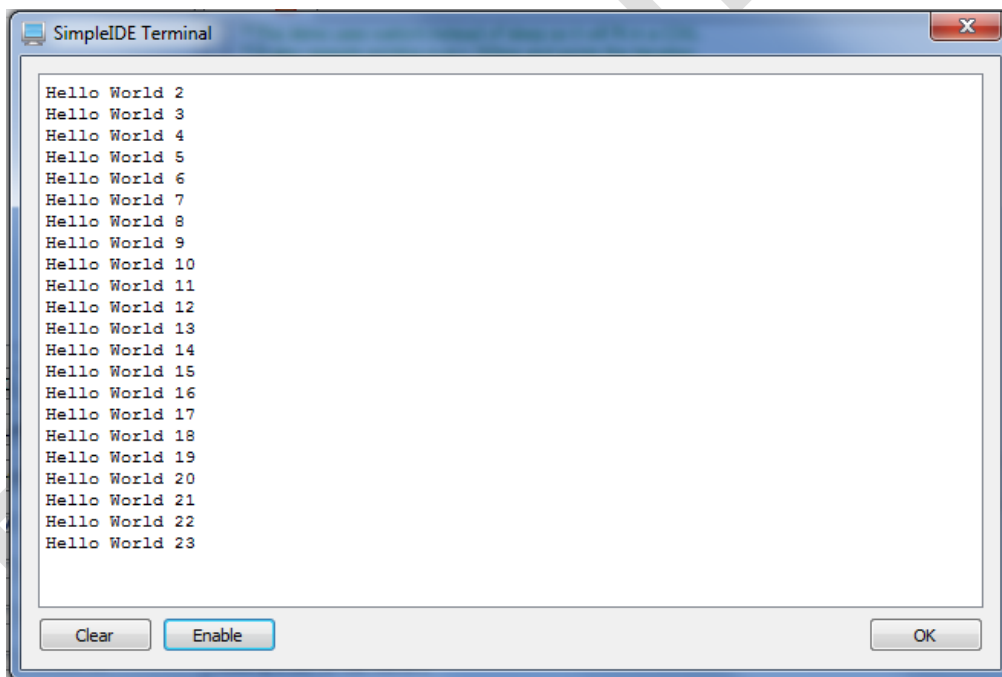
Some computers have serial ports like Bluetooth. Bluetooth serial port programming is possible, but not recommended for new users. Port type can be checked by placing the mouse cursor over the port name without clicking to see "hover help". Make sure the port type show "USB Serial ..." or "FT232 ..."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

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

Running the Hello Demo

1. Open the hello.side project using Menu->Project->Open Project or the tool button. 
2. Use Menu->Run Console F8 or click the blue play button: 

Run Console compiles, loads, and runs the program. The serial console will appear after loading.



Note that on Windows the terminal may not be fast enough to catch the first few output lines. This is because of the transition from propeller-load loading the program, releasing the serial port, and starting the terminal.

Pressing OK or closing the window will close and reset the port. Pressing the Enable/Disable button, will also close the port. The clear button clears the screen.

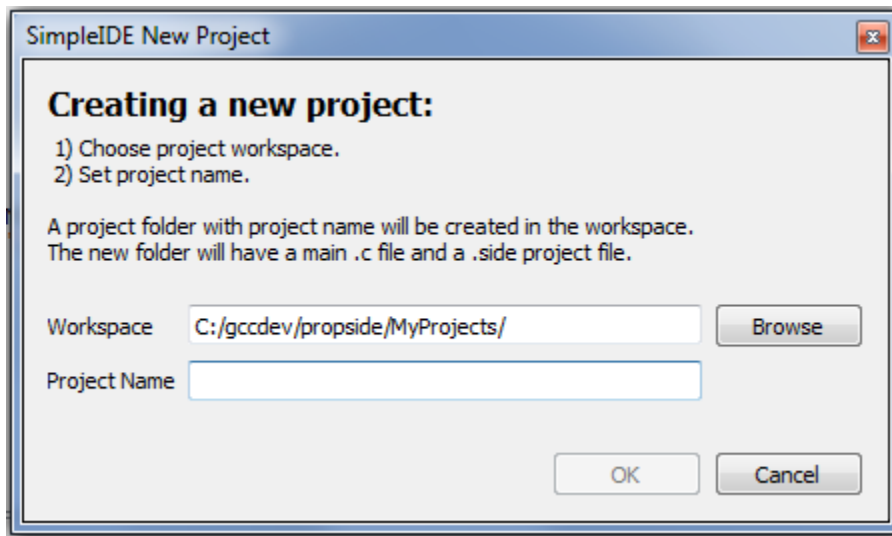
Text can be copied from the terminal using right-click or the OS keyboard copy command.

Creating a new project

There are different ways to start a new project.

Option 1: New Project

1. Choose Menu->Project->New Project or click the New Project button. 




2. Click the folder browse button to select a new folder.
3. Enter the project name. The name will be added to the folder path. When a valid project name and path is set, the OK button will be enabled. A new project can't be started in an existing folder.



Press OK to finish the New Project dialog. The folder/project name will be created. A main program template will be added for the new project.

Option 2: Close All, Use an existing main file

If one has an existing main file it can also be used with SimpleIDE.

1. Close all open files.
2. Choose Menu->Project->Set Project to Current File 
3. Note that project manager only has the existing file listed.
4. Add other files to the project.


Option 3: Close All, start new.


1. Choose Menu->File->Close all to close all editors and the project.
2. Fill in new main program with code below (please do not copy/paste the [code] tags).
3. Choose Menu->File->Save As to rename the "untitled" editor tab to "blinker.c". 
4. Choose Menu->Project->Set Project 
5. Select Memory Model LMM or COG from the project options.

Example Code

```
#include <propeller.h>

int main(int argc, char* argv[])
{
    int mask = 0x3fffffff;
    DIRA = mask;
    for(;;) {
        OUTA ^= DIRA;
        waitcnt(CCLKFREQ/2+CNT);
    }
    return 0;
}
```


Using Menu->Project->Set Project to Current File  will set the project name to the editor tab file name. Set Project saves the project properties to the current state.


Click the hammer button  to build the program after setting the project.

There should be no warnings or errors. If there are warnings or errors in the build status panel, click on the line and it will make the editor zoom to the problem area file and line in the code. Check code for typos.

blinker.c:1:1: error: expected identifier or '(' before '[' token

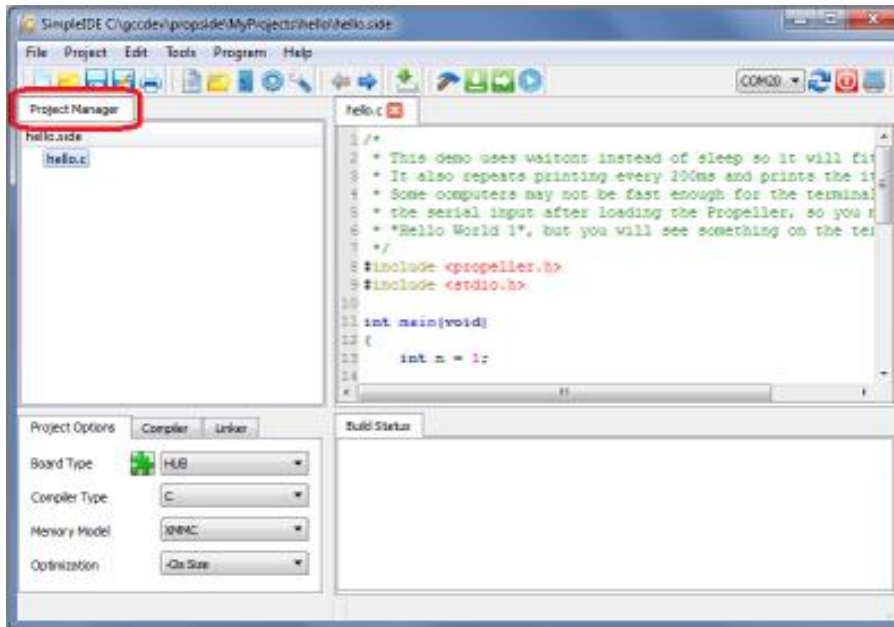
In this example, the compiler found a '[' token in blinker.c on line 1 and column 1.

The program can be downloaded and run with Run (F10) 

Store the program for stand-alone use with Burn EE (F11) 

Project Manager

SimpleIDE uses a project manager - note the tab circled in red below. For C/C++ this does not a list files based on the currently opened and compiled file as in the Propeller Tool. The project manager defines what is built regardless of what is shown in the editor tabs.



A SimpleIDE project is called a "side project" because SIDE is an abbreviation of Simple IDE. The ways to start a side project are described in **Creating a New Project**.

It may be obvious to some, but it is still worth stating that a project consists of several files. If files are not listed in the project, the project can't be built and run. Some .h header files may not be listed, but they must exist in the project folder for a specified link for successful builds.

The minimum required file in the side project is the main .c file. For example, the hello.side project must contain a file called hello.c as shown above. The hello.c file must contain the "main" function, that is: `int main(void)` or some variation of that. Any other files containing code being used by hello.c must be listed in the side project.

Please note, that there is no issue using a main .c type file for projects that include C++ in Propeller-GCC.

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.
- Sometimes we need to specify an include path for the files.
 - An include path can be added with "-I folder" in the Other Compiler Options box
 - An include path can be added using the project manager popup menu.

Object Files

These files are generated by the build process and are not added to the project manager.

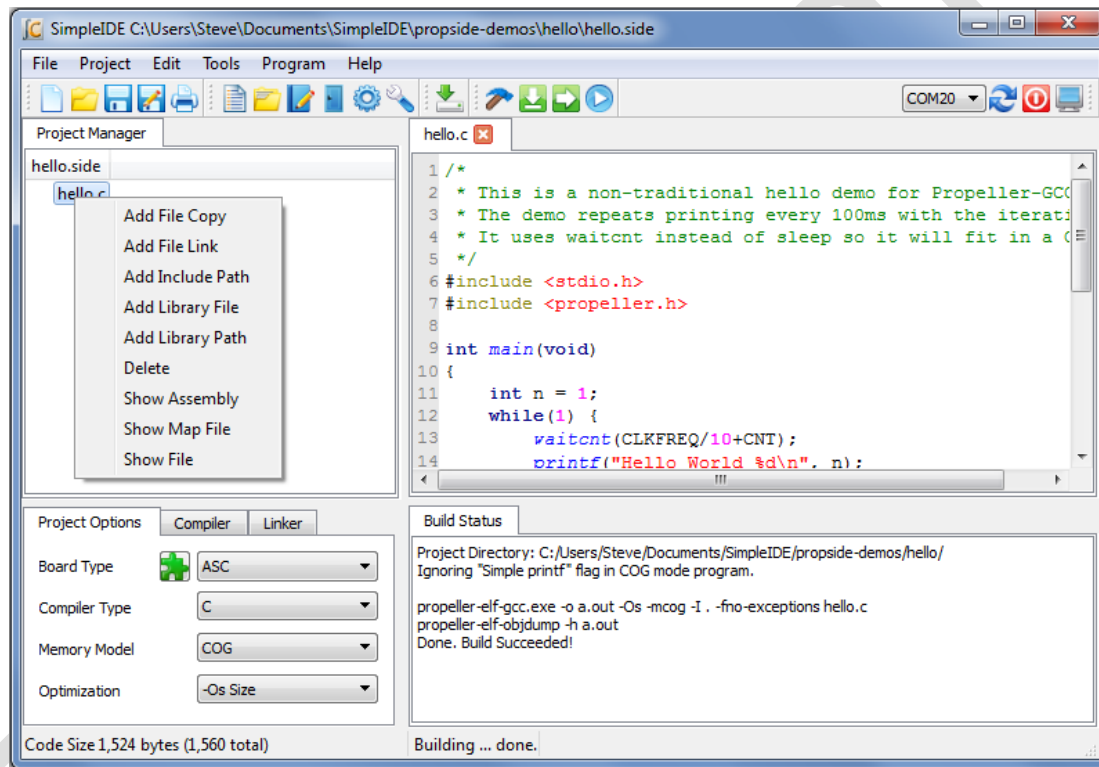
- Object files (*.o): object files are always generated by the compiler.
- COG Object file (*.cog): this is a generated object file created from a .cogc file.
- Dat files (*.dat): this type of file is generated by BSTC or other Spin programs used for making PASM COG code.

Project File List

The project pane is blank if no project is set. If a project is set, the project name with .side will be displayed as in the hello.side example above. All projects are .side projects. The first indented entry is the main program. It should have the main start-up function as in the case of hello.c .

Often we want to have more than one file per project. The project manager lets us add files, links to files, and paths for extra includes and libraries.

Right-click on the hello.c file entry in the project file list (or any other entry in the list), and a menu will pop up that gives several options. Most menu items are related to adding entries like files or links. The function of each popup menu item is described below.



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.

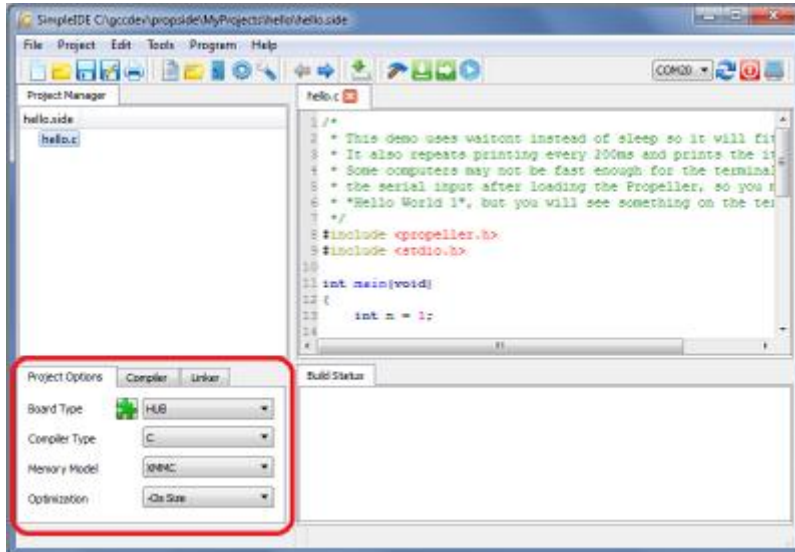
- **Add File Copy:** Add a new file to the project. When adding a file to the project, it must already exist somewhere on the computer. New file can be created in the editor and saved for example. If a file outside of the project directory is selected, the file will be copied to the directory. File names 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 it will not be copied to the project directory. A link will have the short file name and -> full path name as shown above. 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 a -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:** Add a library path for .a library files not in the project folder or tool-chain library folder. This adds a -L path to the project.
- **Delete:** To delete a file or link from the project, right click on the item and choose delete. 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, create it and set the project.
- **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.

Prell

Project Options

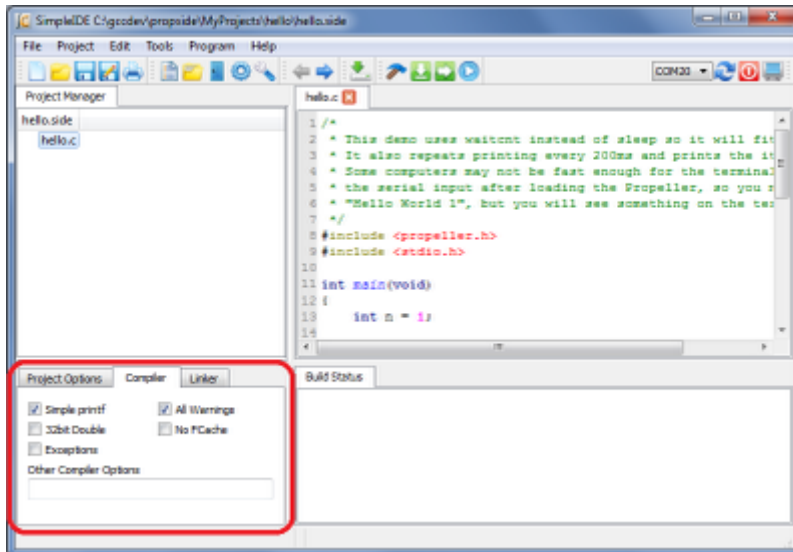
The compiler pane allows choosing typical Propeller-GCC compile options. These options are automatically saved in the .side project file.



- Board Type:** The board type option selects the basic board type where the program will be loaded. In many cases the board type doesn't matter for LMM or COG programs. Some boards have non-standard system clock frequencies though and must be properly selected. In all cases XMM type programs will need an XMM capable board selected.
-  **Reload Board Types:** This puzzle piece button will read all C:\propgcc\propeller-load*.cfg files and load the names into the Board Type combo box. The selected board type is saved in the .side project file.
- Board Type:** This is the drop-down box on the right side of the green puzzle piece. This allows choosing the board type to use with the program. Board types with SDLOAD or SDXMMC are special and when selected tell the IDE that certain functions will be performed.
- Compiler Type:** C compiles C programs and can compile very simple C++ programs such as the C++_toggle demo. Programs needing the std namespace and libraries must use the C++ compiler. Typically a full C++ program will need a large external memory solution to be useful.
- Memory Model:** Many programs use LMM, but some programs use COG or XMMC. The C-VGA demo is a COG only program. To compile that, use the COG memory mode. Some programs are too big for LMM and can be run on external memory modes like XMMC. The included graphics demo can run on LMM, but it can also run on XMMC with board type EEPROM selected (64KB or

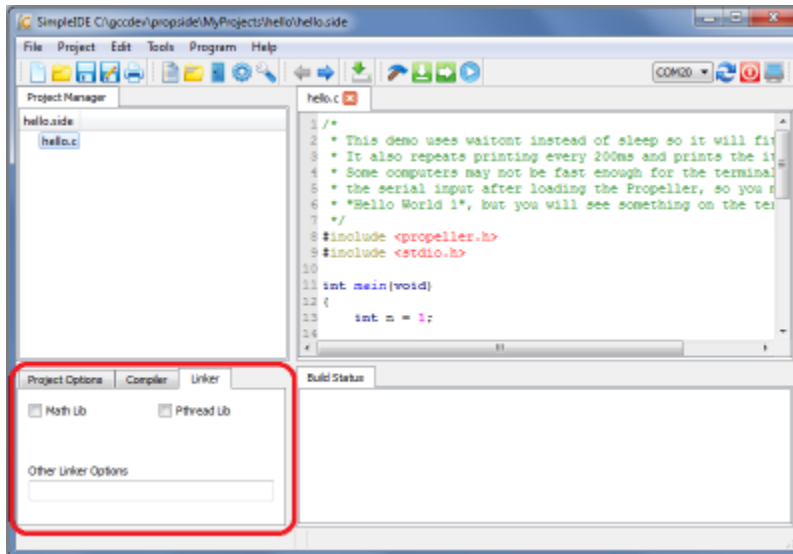
more EEPROM should be used). XMM-SINGLE should be used for RAM based external memory devices. XMM-SPLIT is the same as XMM and should be used where external memory is split into code and data where code lives in Flash and data lives in RAM – XMM is originally used on boards like C3.

- **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.



- **Simple Printf:** GCC's default printf is very big (12KB or more). Simple printf provides most of the default printf features and is fine for most programs although it is not completely standard compliant. Simple printf programs using integers only can be as small as about 7KB so there is great advantage in using it on Propeller most of the time. There are cases though when the normal printf can be slightly smaller than simple printf.
- **32bit Doubles:** Use 32bit doubles for floating point double variables. The default is 64bit 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 bigger programs.

- **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. One may need -I <path-to-library-headers> for using prebuilt libraries.



- **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.
- **Pthread Lib:** 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.
- **Other Linker Options:** This allows adding linker specific options. For example "-l name" may be added for using a prebuilt library. Special linker scripts can be added here if a board does not fit a built-in memory layout.


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.

Special Clock Boards

If the Propeller board does not use 80MHz 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 propgcc\propeller-load\custom.cfg
2. Change clock frequency or clock mode in custom.cfg, and save the file.
3. Click the puzzle piece  in the Project Option control to reload board types.
4. 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 using the serial port console.

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.

- **RCFAST:** The most common of the basic board types are RCSLOW and RCFAST. RCFAST relies on the internal Propeller 12MHz oscillator and will boot any board. Serial port communications with RCSLOW and RCFAST will not behave properly because both modes are imprecise.
- **RCSLOW:** RCSLOW is like RCFAST except that it uses the slowest and most energy efficient clock mode.
- **HUB:** The HUB board type specifies an 80MHz system clock and an external 5MHz crystal with PLL16x clock mode. Any board that has a 5MHz crystal should work with HUB board type. Good serial communications and TV/VGA output are possible with HUB board type and a good 5MHz crystal. Other board types related to HUB are ASC, ASC, QUICKSTART, and others.
- **SPINSTAMP:** The SpinStamp board type is like the HUB board type except that it relies on a 10MHz crystal to produce an 80MHz system clock by using PLL8x in the clock mode. Any Propeller board that has a 10MHz 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 32KB EEPROMs (using two 32KB EEPROMs will not work), 64KB EEPROM, 128KB linear address space EEPROMs (24FC1025 parts do not have a linear address space and will not work. 24LC1024 parts will work), and 256KB EEPROM from ST.

Any set of 64KB+ EEPROM can be configured to add more code space to the program. Using two separate 32KB 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 X* memory model that will work with EEPROM board types. Other board types related to EEPROM that run XMMC programs are 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 (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 (< \$0.75 per MB), low pin count (10 pins for byte-wide solution), high density (up to 32MB), 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 store program code in SPI Flash and data in a device like external SPI SRAM. The only default board type that uses this model today is C3. 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 RUN, RUN Console, or BURN IDE buttons will cause the AUTORUN.PEX output to be downloaded to SD card and then booted to RAM and run. Once the BURN button has been used, the AUTORUN.PEX file can be replaced using the Send File to Target SD Card command/button or by replacing the program with the PC operating system.

SDXMMC Board Types

SDXMMC board types are used to download XMMC memory model AUTORUN.PEX programs to SDCard and run them using the RUN, RUN Console, or BURN buttons. Any board that has an SD card can use SDXMMC.

Preliminary

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

Source Browsing

A source browser is a special IDE feature that allows the user to find symbols in the source files (function declarations and global variable names). For example a complex project can have many files and thousands of lines of code. To simplify dealing with such projects, a program called ctags was created. SimpleIDE uses open source [exuberant ctags](#) to help with source browsing.

In the example below, the function LCD_start is highlighted. This highlight happens when the text cursor is in the the symbol, mouse is over the symbol, Ctrl (or Command on Mac) is pressed, and the symbol is found in the project source files.

The highlight will only appear if the mouse is over a symbol, Ctrl is pressed, and the symbol is found in the project. Library symbols will not be found.

Use the Ctrl+LeftClick (keyboard+mouse) combination to browse to the symbol definition.

Once the function is found, the Go Back button is enabled for browsing back to the previous code. When there is no more "back" left to do, the button goes gray again.

On a PC compatible keyboard, use Alt+RightArrow when the cursor is on a browse-able symbol. Using Alt+RightArrow will find the declaration and Alt+LeftArrow to go back.

One disadvantage of Ctrl browse highlighting is that it can interfere with normal Ctrl-C and Ctrl-V editing a little if the mouse is over a browse-able function or variable name. Just move the mouse if this happens.

Future Improvements

Opportunities for improving SimpleIDE are:

1. Add Spin/PASM project programming and build capability.
2. In the New Project feature the file type to use should be based on the Compiler selected in the Project Options tab.
3. Add a simple built-in browser for allowing F1 Context sensitive help and opening the SimpleIDE User Guide.
4. Add library function and object "dot" lookup to make programming easier.
5. Reduce the number of default configuration files in propeller-load to make the board type list smaller. All configuration files can be put into the share folder.
6. Add pattern matching to Find and Replace.

Other improvements can be requested via email to gccbeta@parallax.com

Miscellaneous Notes

The editor will convert "tabs" to spaces. This choice is based on many years of having to maintain at terribly formatted code. The built-in setting for tabs is 4 spaces. The number of spaces can be set in the General tab of SimpleIDE Properties. Other editors can provide hard tabs for those who need it.

There have been many enhancements and bug fixes since the last update. In the future, bug fixes will be tracked here. For the time being look at propside.googlecode.com for bug fixes.

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>