

PropellerIDE Documentation

Brett Weir

Published
with GitBook



Table of Contents

Introduction	0
User Guide	1
Getting Started	1.1
Installing PropellerIDE	1.1.1
Installing FTDI Drivers	1.1.2
The Main Window	1.2
Tool Bar	1.2.1
Editor View	1.2.2
Code Completion	1.2.2.1
Documentation View	1.2.3
Project View	1.2.4
Keyboard Shortcuts	1.2.5
Project Archive Tool	1.2.6
Debugging Tools	1.3
Terminal	1.3.1
Memory Map	1.3.2
Heat Map	1.3.3
Oscilloscope	1.3.4
Logic Analyzer	1.3.5
Language Support	1.4
Spin	1.4.1
PropBASIC	1.4.2
C	1.4.3
Preferences	1.5
Frequently Asked Questions	1.6
Developer Guide	2
Building PropellerIDE	2.1

Introduction

PropellerIDE is a fun, easy, beautiful editor for the Propeller microcontroller.

- Code the way you like with a colorful, customizable editor.
- Dig deeper into your applications with the built-in memory map.
- Speak your Propeller's language with the integrated serial terminal.
- Find what you need fast with searchable project view and auto-complete.
- Start coding right away with the included Spin Standard Library.
- Runs great on Windows, Mac, Linux, and Raspberry Pi!

User Guide

Getting Started

Installing PropellerIDE

PropellerIDE is currently officially supported on **Windows**, **Mac**, **Debian**, and **Raspbian OS**.

First, [download PropellerIDE](#) for your platform. Then follow the corresponding instructions to get started.

Windows

PropellerIDE is packaged as a Windows installer that will guide the user throughout the installation process.

Mac OS X

PropellerIDE is packaged as a regular DMG image, so mount the Volume and drag the icon into the Applications folder.

Linux

Ubuntu

PropellerIDE requires a minimum of Qt 5.2 which is only available on Ubuntu as of 14.04.

After downloading the Debian package for your platform, install it with `dpkg`.

```
sudo dpkg -i propelleride-(version)-amd64.deb
```

It will complain about dependencies at which point you can run `apt-get` to fix them.

```
sudo apt-get install -f
```

Make sure you install the FTDI drivers!

```
sudo apt-get install libftdi1
```

Add yourself to the `dialout` group so you can use the serial port.

```
sudo usermod -a -G dialout USER_NAME
```

Ubuntu 14.04 or earlier

PropellerIDE is known to build in Ubuntu versions as old as 12.04, but doing so will take some work.

Add the Utopic Unicorn sources to your `/etc/apt/sources.list` .

```
deb http://cz.archive.ubuntu.com/ubuntu utopic main
```

Run an update to ensure your apt repositories are up-to-date.

```
sudo apt-get update
```

Raspberry Pi - Raspbian Wheezy

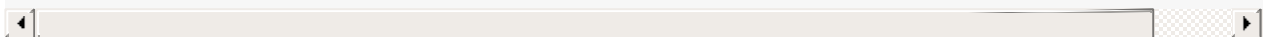
Qt5 is not available in the standard repository, but you can obtain it from Debian backports.

Add the following entries to `/etc/apt/sources.list` .

```
deb http://twolife.be/raspbian/ wheezy main backports
deb-src http://twolife.be/raspbian/ wheezy main backports
```

Add the repository key.

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key 2578B7
```



Update and install Qt5 and its dependencies.

```
sudo apt-get update
sudo apt-get install qt5-default qt5-qmake libegl1-mesa libgles2-me
```



Installing FTDI Drivers

In this tutorial we will walk you through installing the FTDI USB-to-Serial driver. This is required to use the Propeller on all wired platforms.

Windows

Head over to the [FTDI website](#), and scroll down to the chart under "VCP Drivers" and select the download according to your operating system.

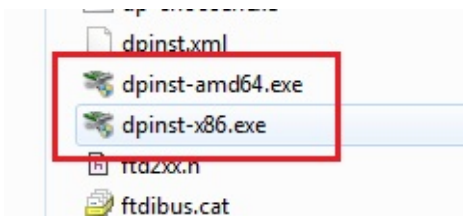
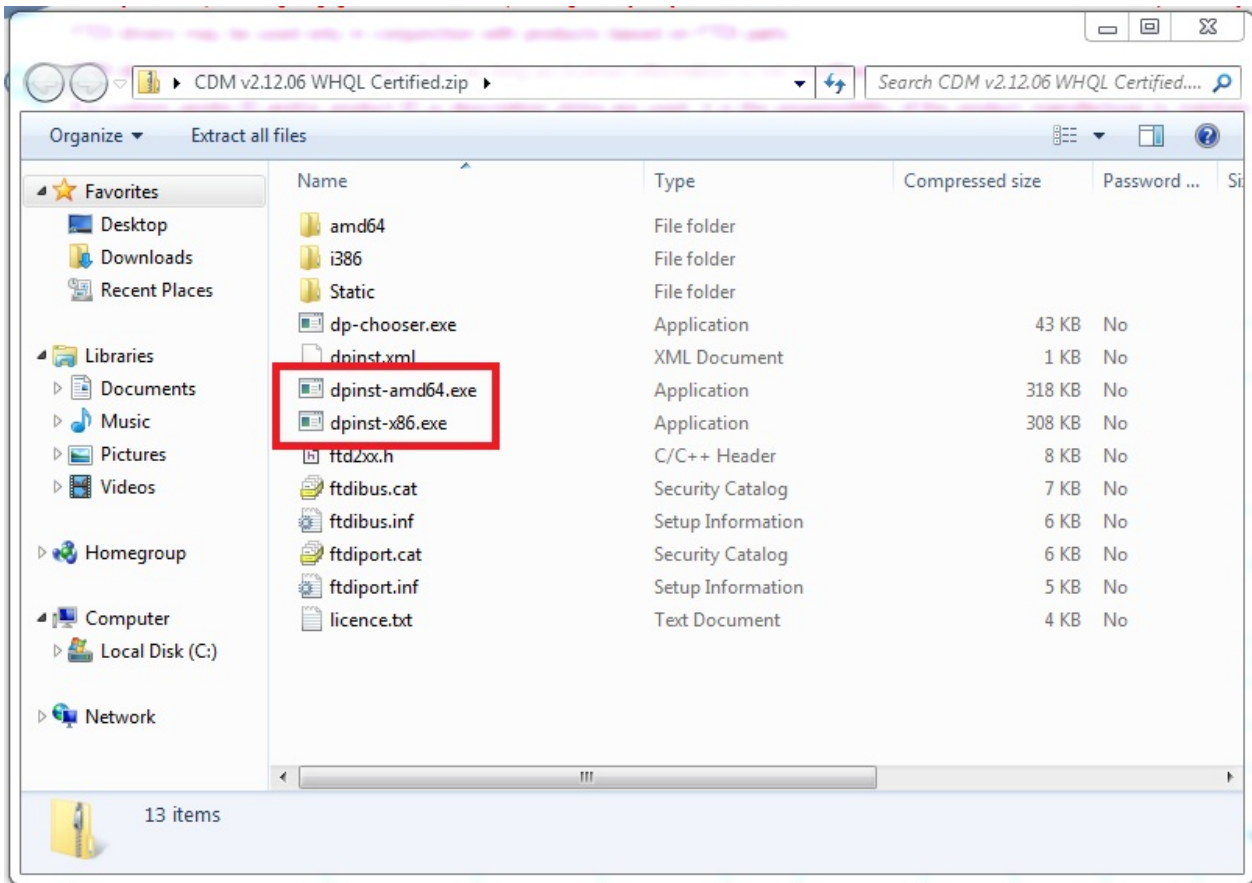
Currently Supported VCP Drivers:

Operating System	Release Date	Processor Architecture							Comments
		x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	
Windows*	2015-07-28	2.12.06	2.12.06	-	-	-	-	-	2.12.06 WHQL Certified Available as setup executable Release Notes
Linux	2009-05-14	1.5.0	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19 Refer to TN-101 if you need a custom VCP VID/PID in Linux
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS
Mac OS X 10.9 and above	2015-04-15	-	2.3	-	-	-	-	-	This driver is signed by Apple
Windows CE 4.2-5.2**	2012-01-06	1.1.0.20	-	-	1.1.0.20	1.1.0.10	1.1.0.10	1.1.0.10	
Windows CE 6.0/7.0	2012-01-06	1.1.0.20 CE 6.0 CAT CE 7.0 CAT	-	-	1.1.0.20 CE 6.0 CAT CE 7.0 CAT	1.1.0.10	1.1.0.10	1.1.0.10	For use of the CAT files supplied for ARM and x86 builds refer to AN_319
Windows CE 2013	2015-03-06	BETA			BETA				BETA VCP Driver Support for WinCE2013

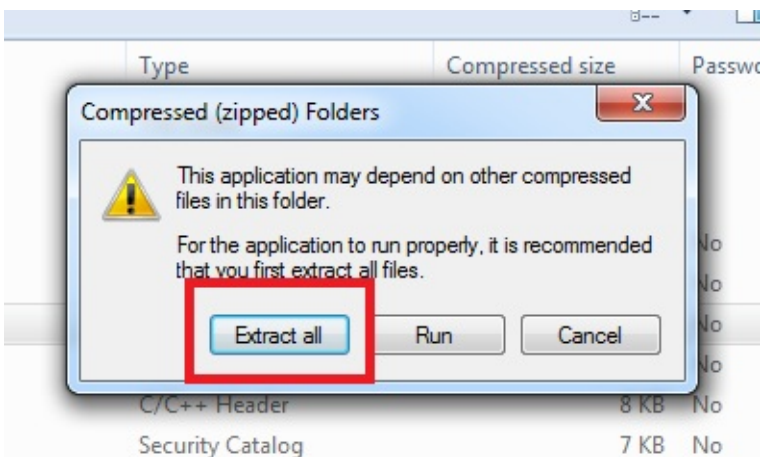
Select the most recent version and start the download (either will download the same zipped file).

Operating System	Release Date	Processor Architecture		Pr
		x86 (32-bit)	x64 (64-bit)	
Windows*	2015-07-28	2.12.06	2.12.06	F

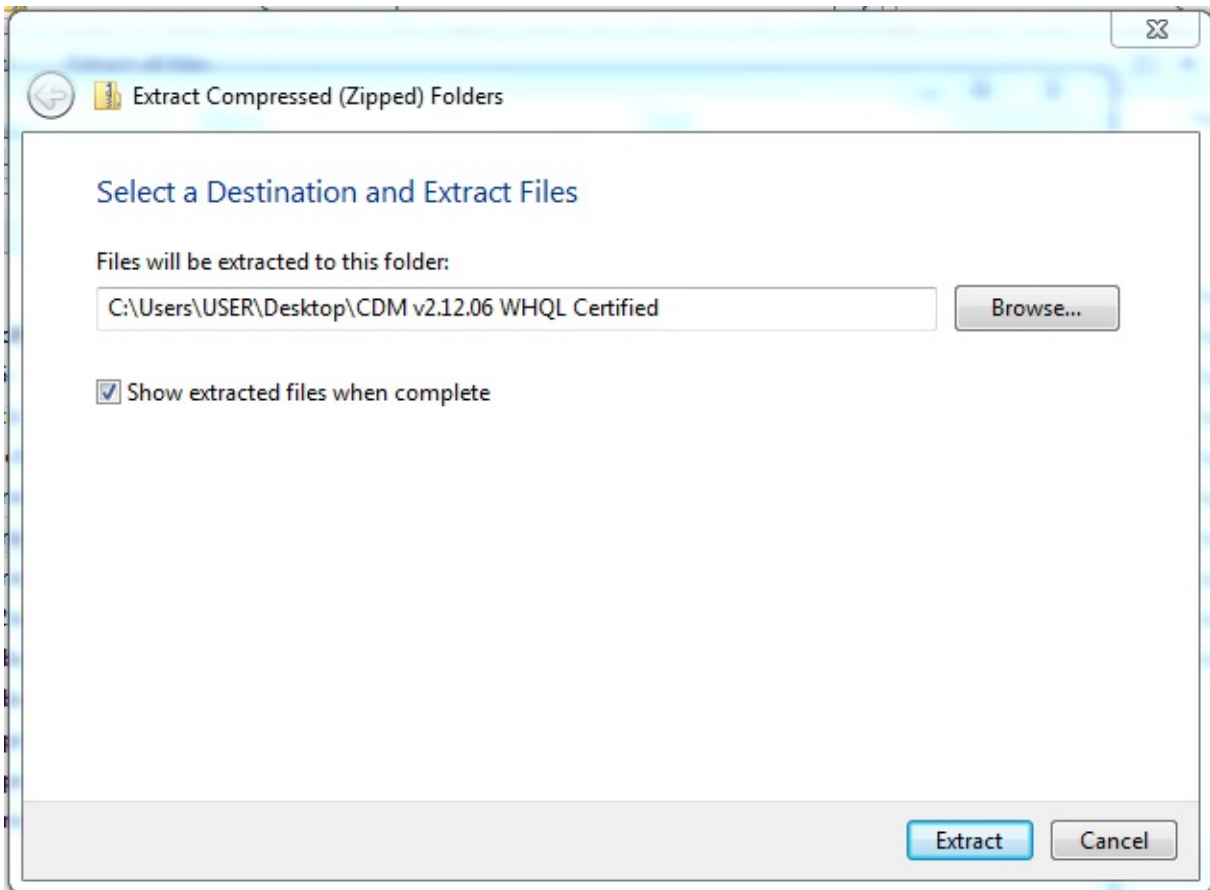
After download finishes, open the zipped file. This file will contain both executables for 32- and 64-bit processors. Select the one your computer runs (dpinst-amd64 for 64-bit and dpinst-x86 for 32-bit).



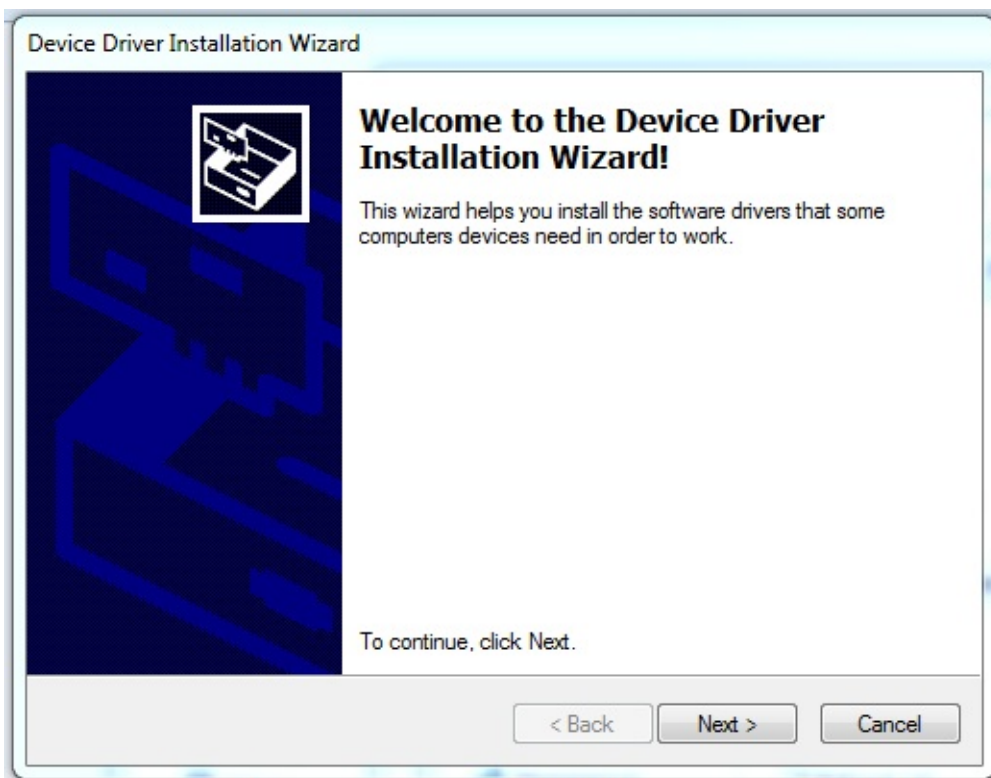
Click "Extract All" when a window pops up to extract the files.

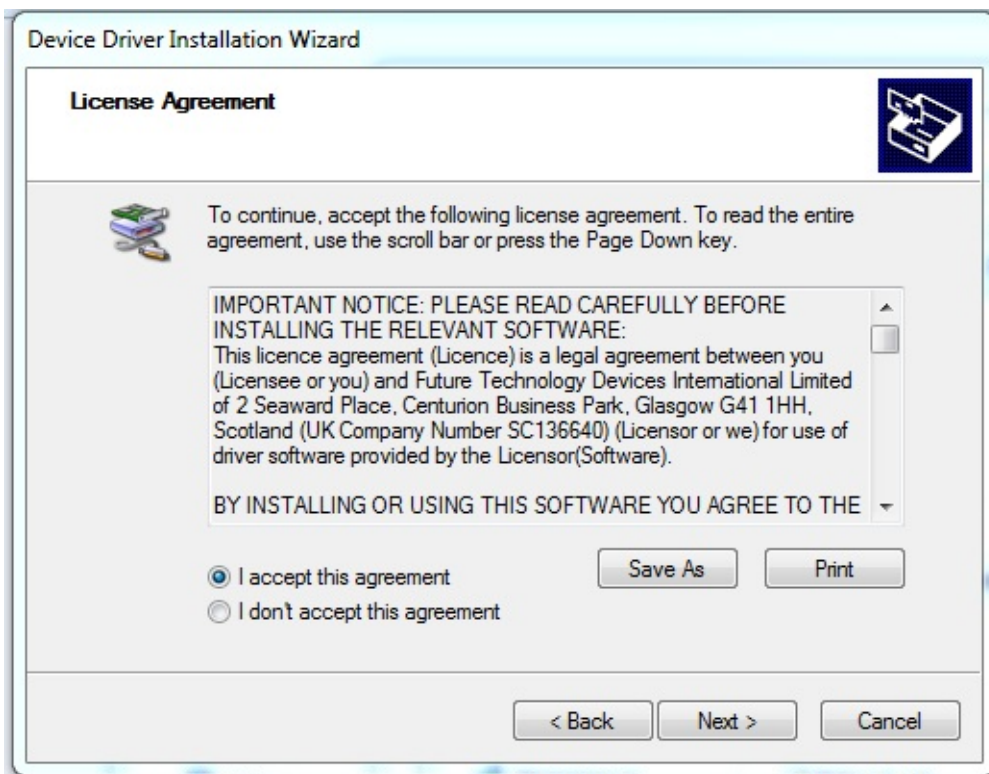


Click "Extract" into a destination. Another window will pop up containing files.



In this popped up window, double click on the same executable from before. Remember to click ""Yes" to allow the program to make changes to your computer.





Follow the instructions until installation is completed.

Linux

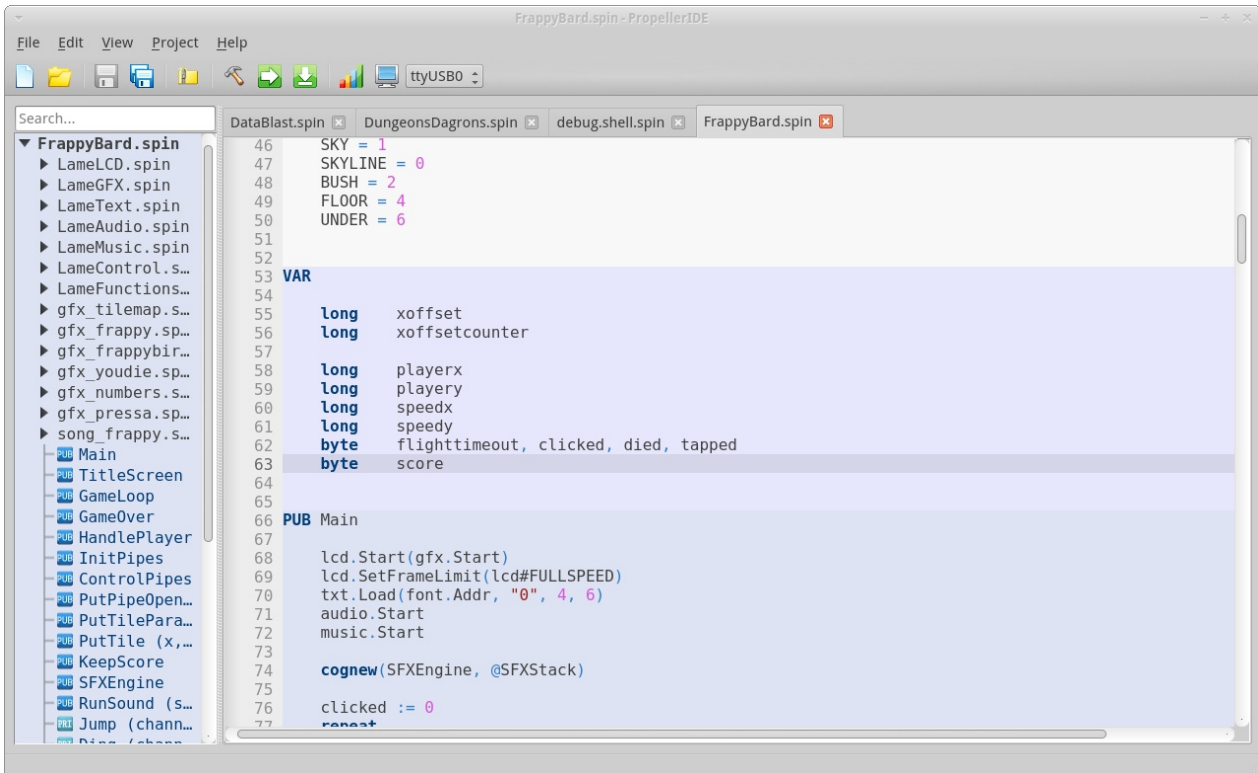
On Debian, open a terminal and use the "apt" package manager to install to your system.

```
sudo apt-get install libftdi1
```

On Fedora, use the "yum" package manager.

```
sudo yum install libftdi
```

The Main Window



Tool Bar



File

New

Creates a new file.

Open

Opens an existing file.

Save

Saves the current file to disk.

Save As

Saves a copy of the current file under a new name. The original file is not saved.

Project

Archive Project

Builds an [archive](#) of the current project.

Build Project

Compiles the current project without downloading it.

Run Project

Compiles and downloads it to the currently selected board.

Write Project

Compiles the project, downloads it to the board, and writes it to the board's first EEPROM.

Device Selector

Selects the target download device from any available devices connected to the system.

Debugging

Memory Map

Opens the [Memory Map](#) widget.

Terminal

Opens the [Terminal](#) widget.

Editor View



The screenshot shows the PropellerIDE Editor View with five tabs: DataBlast.spin, DungeonsDagrons.spin, debug.shell.spin, FrappyBard.spin, and com.serial.terminal.spin. The active file, FrappyBard.spin, contains the following code:

```
1 {{
2 |
3 |
4 |
5 |
6 |
7 |
8 |
9 |
10 |-----|
11 | A shockingly original game for the LameStation!
12 |-----|
13 | Brett Weir, 2014
14 |}}
15
16 CON
17   _clkmode = XTAL1 | PLL16X
18   _xinfreq = 5_000_000
19
20 OBJ
21
22   lcd      : "LameLCD"
23   gfx      : "LameGFX"
24   txt      : "LameText"
25   audio    : "LameAudio"
26   music    : "LameMusic"
```


Code Completion

With PropellerIDE, you can complete code from the current file or another file in the path in the main file's directory or a directory in the library path.

The following completions are supported.

- Public functions
- Constants

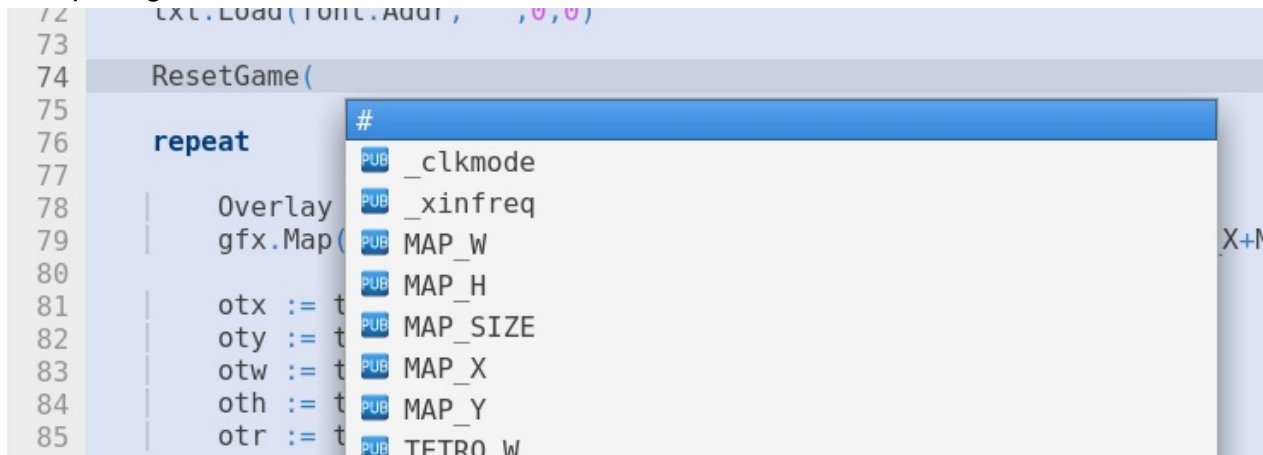
Completing From The Current File

As of v0.36.7, PropellerIDE supports completing within the current file.

- Typing `#` brings up a list of constants.
- Typing `.` brings up a list of functions.

Press Esc to quit without completing. Press Enter or Return to accept the completion.

Completing a constant from the current file.



Completing From Another File

You can also complete code from another file.

- Typing the alias of an object[1], then `#` opens a list of constants.
- Typing the alias of an object, then `.` opens a list of functions.

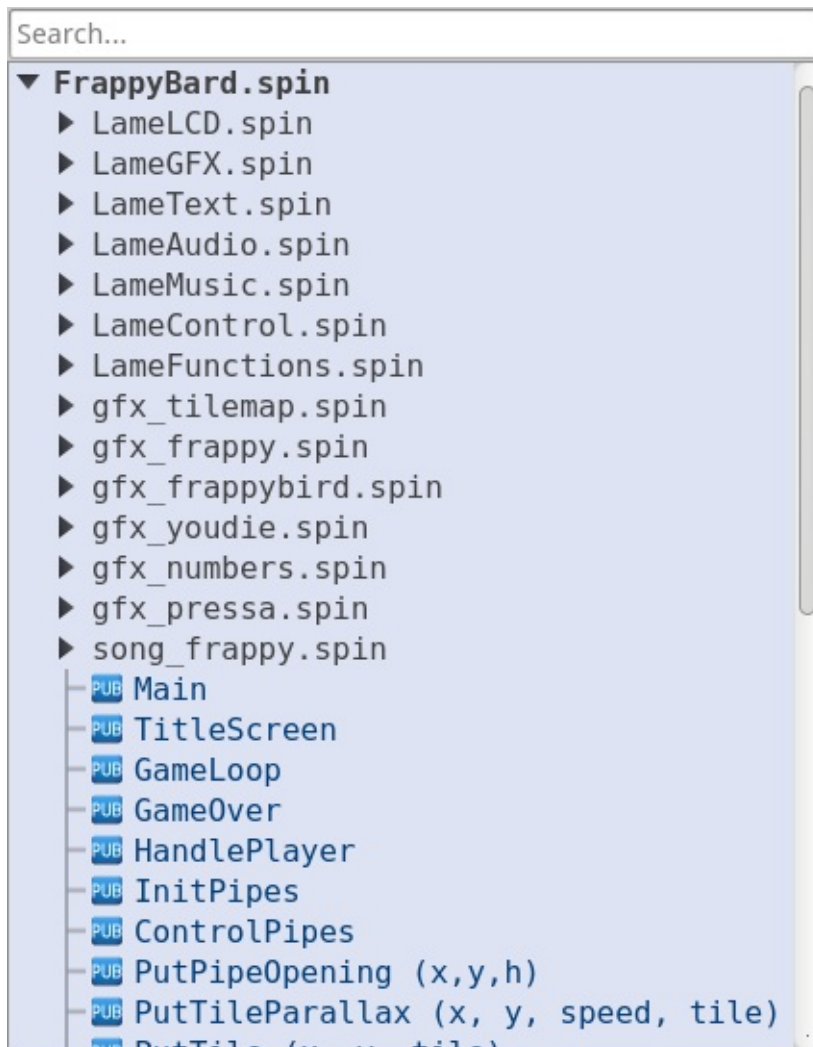
As with before, Esc quits, Enter or Return accepts.

Completing a function from another file.



1 e.g. where we declare an object `kbd : "Keyboard"` , the alias is `kbd` .

Project View



Keyboard Shortcuts

Basic Shortcuts

Editing Shortcuts

Ctrl+Z	Undo
Ctrl+Shift+Z	Redo
Ctrl+X	Cut
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+A	Select all

View Controls

Ctrl++	Increase font size
Ctrl+-	Reduce font size
Ctrl+X	Cut
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+A	Select all

Project Controls

F8	Open memory map
F9	Compile current program
F10	Run current file
F11	Write current file
F12	Open terminal on current device

Tab Controls

Ctrl+T	Create a new file
Ctrl+Shift+T	Create a new file from a template
Ctrl+W	Close the current tab
Ctrl+PgUp	Go to previous tab
Ctrl+PgDn	Go to next tab

Debugging Tools

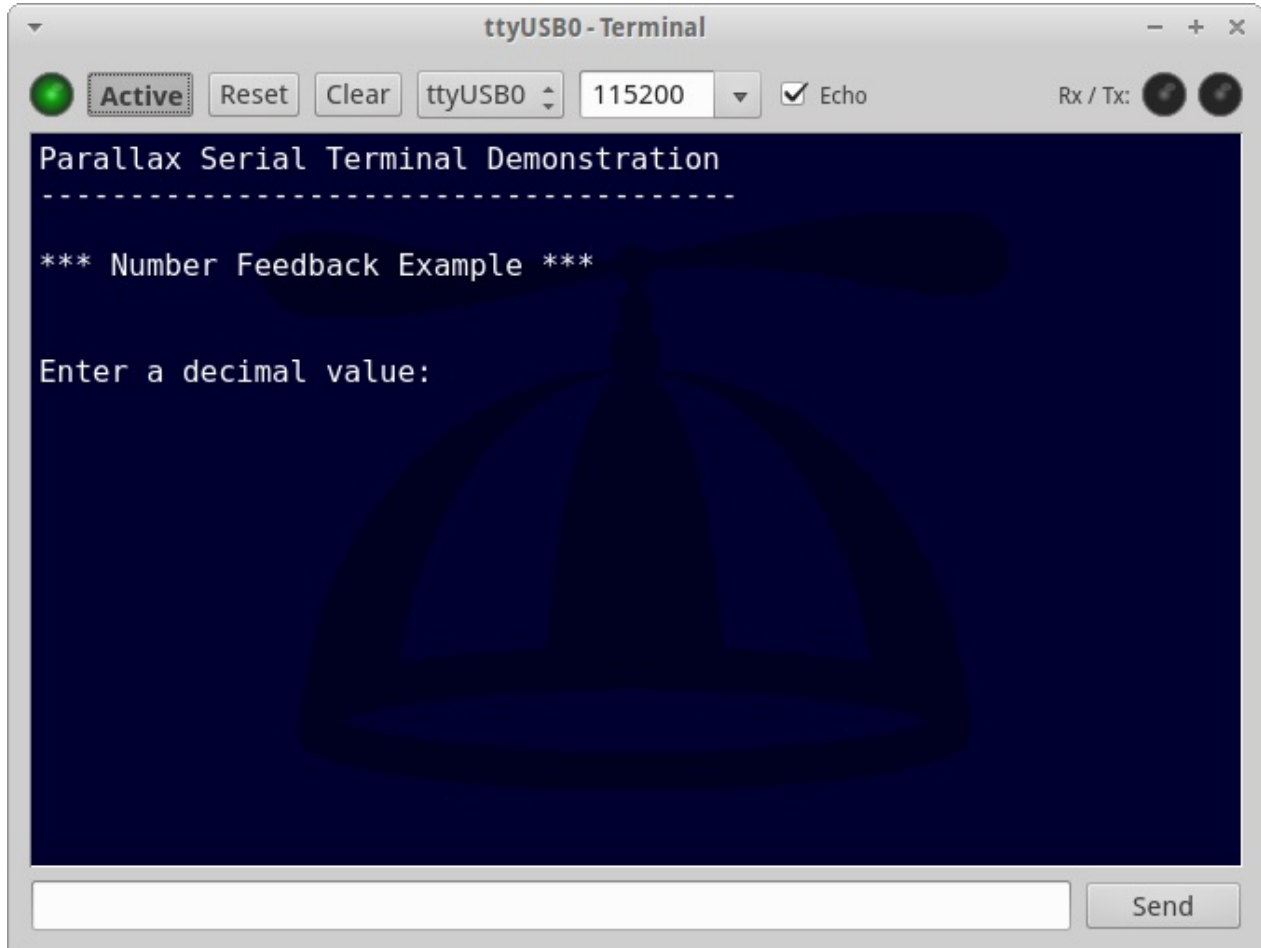
You just wrote an awesome program, you sit down to test it out, you cross your fingers, dripping with anticipation, and... it doesn't work.

Agh! What a frustrating moment!

Luckily, PropellerIDE can help. It has lots of built-in tools to help you pinpoint where you went wrong.

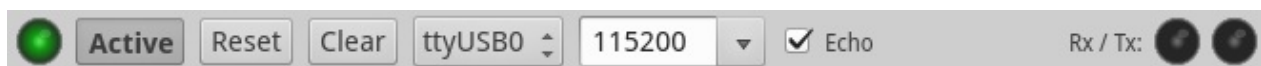
Terminal

PropellerIDE has a built-in serial terminal you can use to debug your Propeller programs.



There are no restrictions on the number of terminals that can be open at a time, even on the same device, and software can be downloaded to attached devices without disconnecting terminals first.

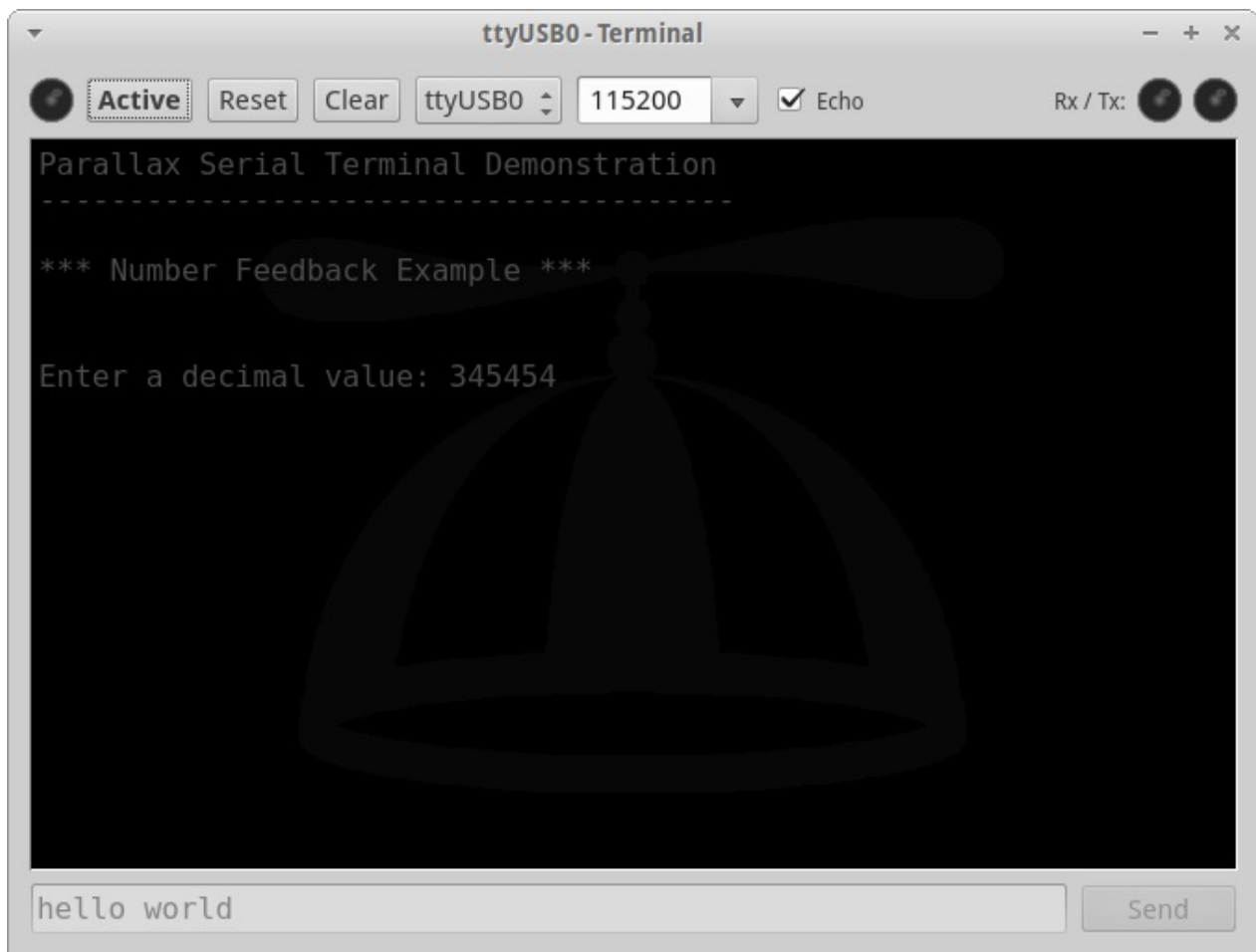
Tool Bar



Active Active Light & Button

When the light is green and **Active** is pressed, the device is connected and ready to send and receive data.

When the terminal is disconnected, it goes black and will stop receiving data and responding to key presses.



Reset

Sends a hardware reset to the board.

Clear

Clears all text from the console and sets the cursor back to the top left.

Device

The name of the device this terminal is currently attached to. This list of devices varies depending on your platform, but generally speaking, they look as follows:

Serial devices:

Windows	COM1, COM2, ...
Linux	ttyUSB0, ttyUSB1, ...
Mac	cu.usbserial-...

Wifi devices:

not yet available

Baud Rate

The rate of transmission to the board. Type in the baud rate you want, or click the arrow to select from the following baud rates:

9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600

The default baud rate is `115200`.

Echo **Echo**

When echo is enabled, everything you type will be copied to the console, in addition to being sent to the device. Some software expects echo to be enabled, while others send the data back to the console themselves, which will result in duplicated text.

Try toggling this feature if your application isn't behaving how you want it to.

Rx / Tx Lights

These lights indicate when data is received or sent. Red is received, blue is sent.

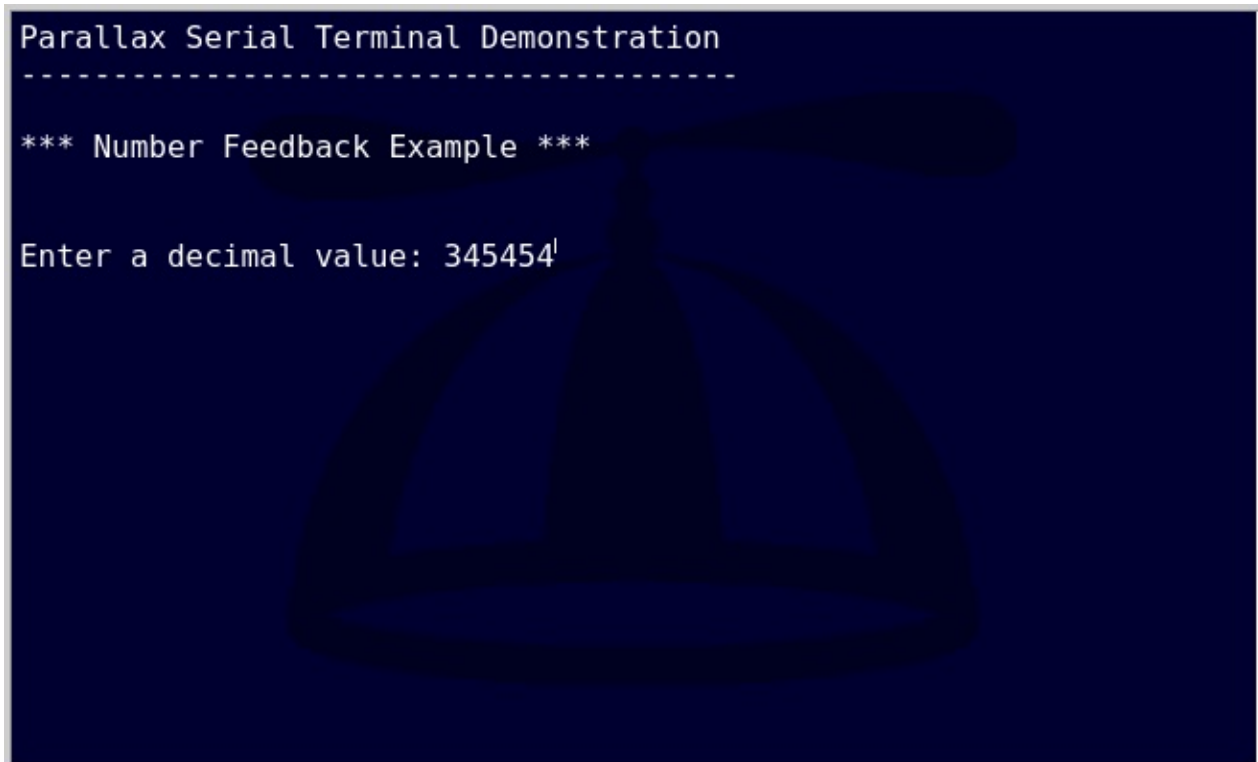
Input

There are two ways to input text

- Through the console itself
- Through the *input line* at the bottom of the window

Console

Using the console is recommended when the target device supports a more advanced command-line interface, as it will allow you to take advantage of things like readline capabilities, cursor positioning, etc.



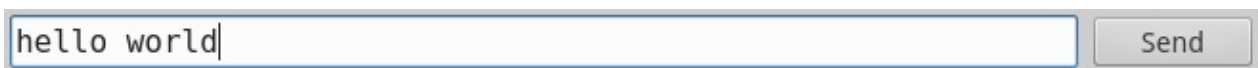
```
Parallax Serial Terminal Demonstration
-----
*** Number Feedback Example ***

Enter a decimal value: 345454'
```

Pressing the `Enter` or `Return` key sends a single newline (ASCII `10`) to the device.

Input Line

For simpler interfaces, the input line is a better choice.



Pressing the `Enter` or `Return` sends the text without newline. Pressing the `Send` button sends the text plus a single newline.

Parallax Serial Terminal Compatibility

The following ASCII characters implement basic terminal compatible with the original Propeller Tool's serial terminal.

16	Clear Screen
11	Clear to End of line
1	Home cursor
2	Position Cursor in x,y
14	Position cursor in X
15	Position cursor in Y
13	New Line
10	Line Feed
3	Move cursor Left
4	Move cursor Right
5	Move cursor Up
6	Move cursor Down
9	Tab
8	Backspace

PropellerIDE includes a corresponding `com.serial.terminal` object that implements these control characters.

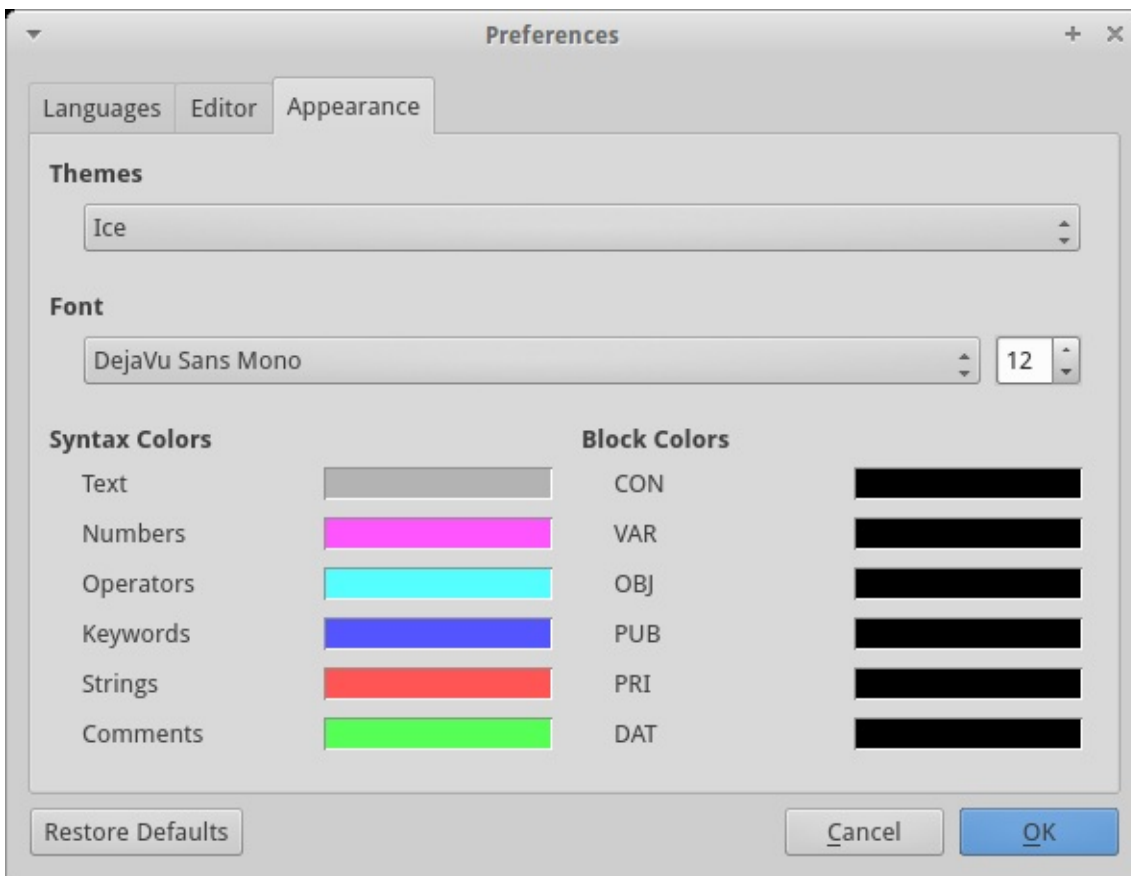
Language Support

Spin

Preferences

Appearance

PropellerIDE can be themed to suit your preference in the *Appearance* tab. Click to select from a drop-down of themes or monospace fonts, or double-click on one of the color swatches to open a color picker.

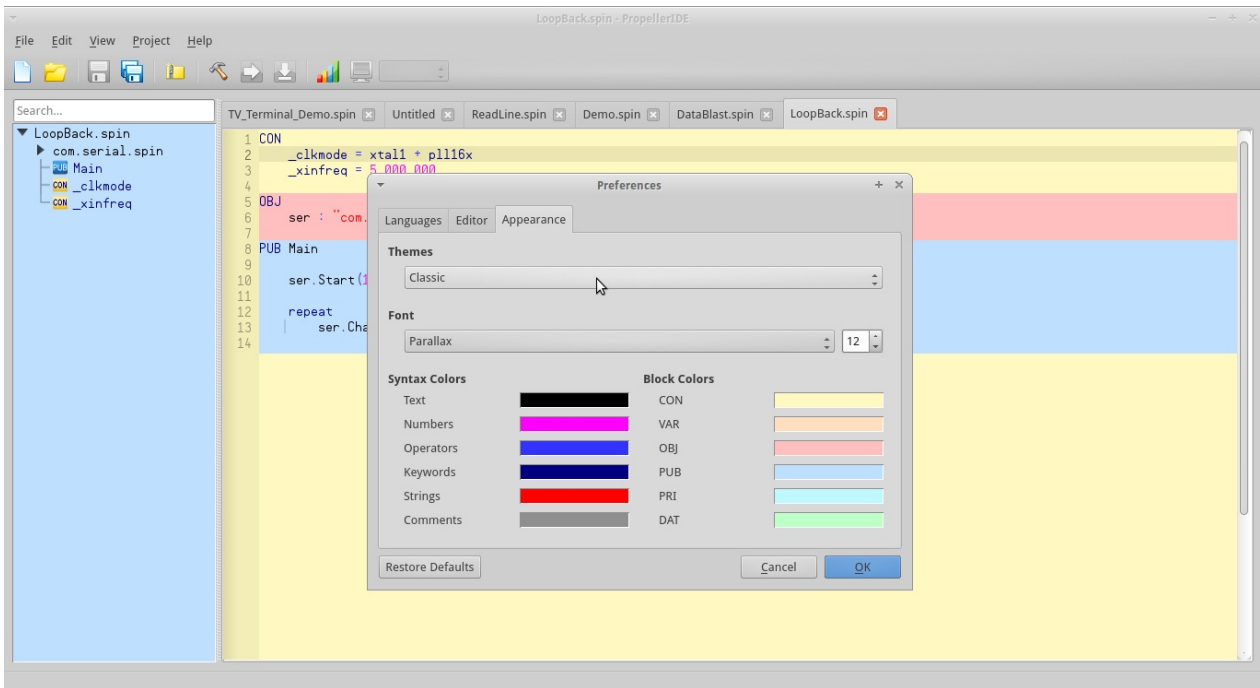


Changes to the appearance will propagate instantly throughout the IDE.

Classic Theme

PropellerIDE provides the *Classic* theme for compatibility with the original Propeller Tool. It supports the legacy Parallax font with the proprietary Propeller character mapping.

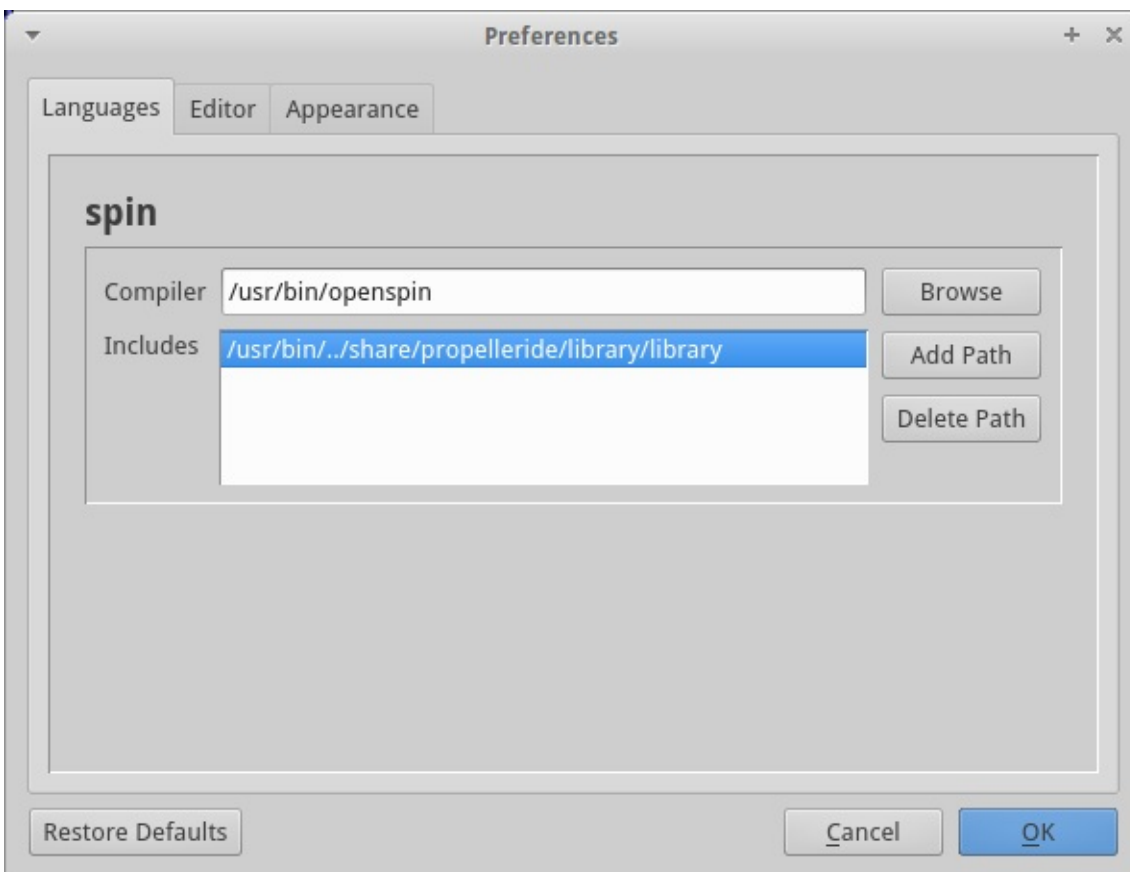
Warning	The Parallax font is deprecated The proprietary character mapping is not portable and is only provided for legacy support. Consider the use of a standalone diagramming tool or plain ASCII for creating in-source diagrams.
---------	---



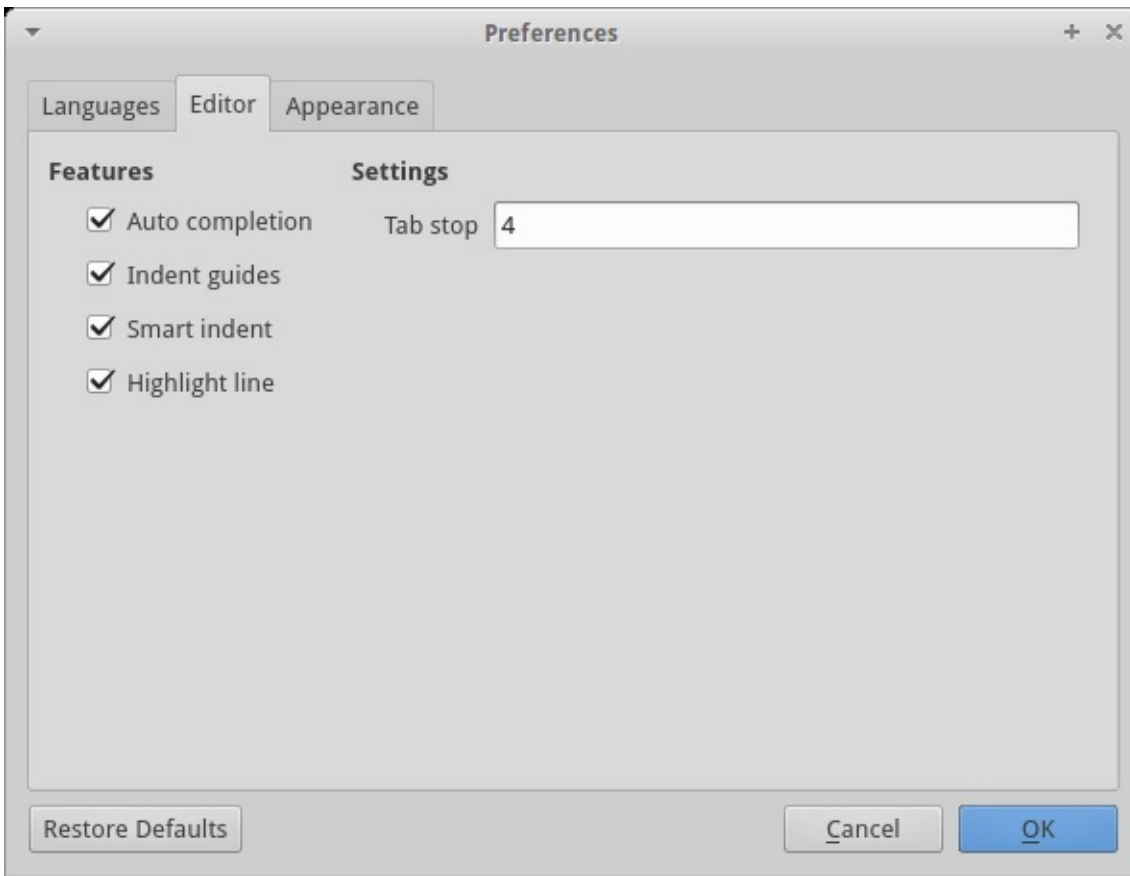
Languages

PropellerIDE supports multiple library paths to be searched from top to bottom.

Note	This is still an in-progress feature.
------	---------------------------------------



Editor



Auto completion

Toggles code completion

Frequently Asked Questions

1. *What is PropellerIDE?*

A modern editor for the Parallax Propeller.

Developer Guide

Building PropellerIDE

The following dependencies are needed to build PropellerIDE:

- Qt5.3 or later

PropellerIDE has been built on the following platforms:

- Windows (Vista, 7, 8)
- Mac OS X (10.6 onward)
- Ubuntu (12.04 onward)
- Raspbian OS
- pcDuino

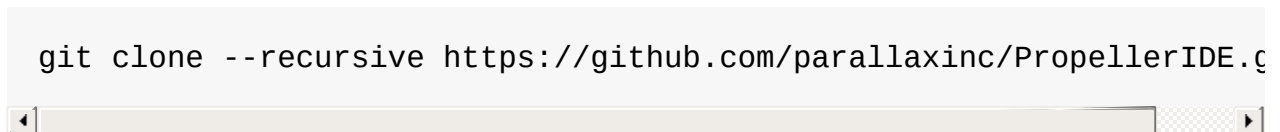
PropellerIDE has been built with the following compiler toolchains:

- GCC
- MinGW-x64
- Clang
- MSVC

Getting the source

Check out the project and its dependencies.

```
git clone --recursive https://github.com/parallaxinc/PropellerIDE.c
```



Building the executable

Using `qmake`

PropellerIDE can be built from the command-line using `qmake` to generate makefiles.

```
cd PropellerIDE
qmake
```

If you have made changes to the `.pro` files, remember to use `-r` to update all makefiles, not just the current one.

```
qmake -r
```

Use Make to build the project. On most Linuxes, GNU Make is ubiquitous. PropellerIDE supports parallel builds, so make sure to specify the number of jobs with `-j`.

```
make -j16
```

The makefiles support standard makefile targets: `make clean` removes object files, `make distclean` removes object files and makefiles.

Windows

You will need to download Qt5 from the Qt website. You will also need Inno Setup to build the Windows installer.

- <https://www.qt.io/download/>
- <http://www.jrsoftware.org/isinfo.php>

Qt is distributed with either a MinGW or MSVC toolchain on Windows. Be sure to add the paths to the toolchain and Inno Setup to the system environment.

```
C:\Qt\Tools\mingw482_32\bin;C:\Qt\5.3\mingw482_32\bin;C:\Program Fi
```

The MinGW toolchain is painfully slow, but you can build with `mingw32-make`, which supposedly supports parallel builds but is slow as a dog anyway so it makes little difference.

```
mingw32-make
```

If you've decided to install Visual Studio, you'll have an instance of `nmake`. You can enable parallel builds by setting the CL environment variable, which will speed things up considerably.

```
set CL=/MP
```

Then start the build.

```
nmake
```

Using QtCreator

PropellerIDE may also be built with QtCreator, but it should be noted that QtCreator and `qmake` builds seem to be incompatible with each other, so `make distclean` should be called before switching between them.

Using CMake

Instructions on CMake builds are not yet available.

Packaging For release

PropellerIDE is distributed in standalone packages using `packthing`, an open source packaging tool available on [GitHub](#) or downloadable via the Python Package Index.

Via GitHub:

```
git clone https://github.com/lamestation/packthing
cd packthing
pip install -r requirements.txt
python setup.py install
```

Via PyPI:

```
pip install packthing
```