Parallax Propellent

The Parallax Propellent software is a Windows-based tool for compiling and downloading to the Parallax Propeller chip. Propellent is available as both a library (Propellent.dll) and as an executable (Propellent.exe).

- The Propellent Library (DLL) is for software developers to link into applications enabling immediate support of the Propeller using the same functions as the Parallax-made Propeller Tool development software.
- The Propellent Executable (EXE) is a program that includes the Propellent Library within it and provides many of the same functions to anyone wishing for command-line support of the Propeller chip.

This document is written for the Propellent Library. For details about the Propellent Executable, see the "Propellent Executable.pdf" document, or run the executable with the '/help' switch. For more information on the Parallax Propeller chip and tools, please visit http://www.parallax.com/propeller

Features of the Propellent Library:

- Can download Propeller Application images (.binary or .eeprom) to Propeller chips.
- Can compile Propeller source (.spin) for downloading, for saving as a binary or eeprom image, for retrieval of source document, or just for syntax checking.
- Can run with full, limited, or no visual feedback.
- Includes the Propeller Tool's multi-threaded serial port handling and Propeller chip communication functionality.
- Includes the Propeller Tool's dialogs for indicating serial port access and download progress as well as the user-customizable serial port search options.
- Automatically stores user-modified preferences in the Windows Registry for use in future sessions. This can be disabled if desired.
- Allows access to current list of available serial ports.
- Allows calling application to view and/or specify preferences such as Library Path, Reset Signal (DTR, RTS, or both) and Serial Search Method (AUTO or specific port).
- Supports Win2K (and later) operating systems.

New in Propellent Library v1.5

For users of the previous Propellent Library (v1.4):

1) Fixed bug in Propellent Library v1.5 that output or displayed a source line from the wrong file when a compile error was caused by a missing object file.
2) Enhanced dialog display to make source line, and error message, more clear.

For users of the previous Propellent Library (v1.3):

1) Propellent Library v1.4 now includes a more robust handshake algorithm (like that of Propeller Tool v1.3.2) for connecting with boards like the Propeller QuickStart board.

For users of the previous Propellent Library (v1.2):

1) Fixed bug in SetSerialSearchRules that prevented the rule strings from being properly validated, cleaned, and updated.

For users of the previous Propellent Library (v1.1):

1) GetPropellerVersion has been updated to return a PChar instead of a LongWord. See details below.
2) New routines, GetGUIMode and SetGUIMode, allow manipulation of the amount of GUI-based messages that display during normal operation. See details below.
3) SaveImage routine now automatically creates any non-existing folders listed in the Filename path. See details below.

Propellent Library API

To use the Propellent Library (Propellent.dll):
1) Store the Propellent.dll file in your application's path.
2) Link the Propellent.dll library into your application either statically or dynamically, as desired, using whatever means required by your development tools.
3) Call InitPropellent to initialize internal library settings and specify options.
4) Call other routines as desired to get/set options, compile, find a Propeller chip and download to it.
5) Call FinalizePropellent, before unloading library or terminating application, to clean up settings and free memory used by the Propellent Library.

Important:
To Compile (with the CompileSource call; see below) the library folder path must first be set either by a previous session or by a call to SetLibraryPath.

Library API:
Calling Conventions: All library routines should be called using "cdecl" format.

Types: Boolean = True/False.
Byte = 8-bit integer.
LongWord = 32-bit unsigned integer or pointer.
PChar = 32-bit pointer to zero-terminated character string.
Pointer = 32-bit pointer to data.

Routine (*parameters*) | **Returns** | **Description**

InitPropellent (*WinHandle*, *StorePrefs*, *RegPath*) | nothing

REQUIRED. This routine initializes critical internal items for proper library function. This MUST be the first call made to the Propellent Library and should be done only once per library loading (if dynamically linked) or application initialization (if statically linked).

WinHandle (LongWord) : Handle to calling application's main window.

StorePrefs (Boolean) : True = Propellent library automatically stores its preference items in the Windows registry for use by future sessions. (See *RegPath* for default path). False = Propellent library does not save its settings; calling application is responsible for maintaining preference items.

RegPath (PChar) : May be null (default) or a pointer to a string indicating the registry path in which to automatically store preference items. This value is ignored if StorePrefs = False.

Set *RegPath* to null to use the default path. Set *RegPath* to the address of a path string to have the library automatically store its settings in another path (such as your application's registry path); a new key will automatically be appended to the path.

The default path is <rootkey>\SOFTWARE\ParallaxInc\Propeller\Propellent\<lib_reg_ver> where <rootkey> is HKEY_CURRENT_USER, and <lib_reg_ver> is the library's registry version (ex: 1.0).

If not null, path must be a valid path within the HKEY_CURRENT_USER hive; do not include "HKEY_CURRENT_USER\" in the path. If it doesn't already exist, the path plus the <lib_reg_key> will be created automatically at the end of a session if a setting had been modified within that session.

GetLibraryVersion	LongWord	**OPTIONAL.** Call this to retrieve the Propellent Library's version in Binary Coded Decimal (BCD) #.# format. For example, a returned value of $10 means version 1.0.
GetResetSignal	Byte	**OPTIONAL.** Get Propeller chip reset signal preference. Returns 0, 1, or 2; 0 = toggle DTR (default), 1 = toggle RTS, 2 = toggle both DTR and RTS. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True.
SetResetSignal (*Value*)	nothing	**OPTIONAL.** Set Propeller chip reset signal preference. The default (DTR) will work for all true serial ports and for most USB-to-Serial devices, however some USB-to-Serial devices only support RTS (not DTR). This option allows the user to set which pin (DTR, RTS, or both) is used to reset the Propeller chip at the start of the identification or download process. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True. *Value* (Byte) : 0 = toggle DTR (default), 1 = toggle RTS, 2 = toggle both DTR and RTS.
GetGUIMode	Byte	**OPTIONAL.** Get Propellent Graphical User Interface preference. Returns 0, 1, 2, or 3; 0 = all on (default), 1 = status messages only, 2 = dialogs only, 3 = all off. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True.
SetGUIMode(*Value*)	nothing	**OPTIONAL.** Set Propellent Graphical User Interface preference. This option allows the user to specify how many GUI-based messages should be displayed during normal operation; all messages, status (progress messages) only, dialogs (user prompts) only, or none. Note that this setting is ignored for: 1) the ShowEditPorts function since that feature requires GUI interaction, and 2) any rare-occurrence user prompts such as a "File already exists. Overwrite?" dialog. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True. *Value* (Byte) : 0 = all on (default), 1 = show status (i.e.: progress messages) only, 2 = show dialogs (i.e.: user prompts) only, 3 = all off.
GetSerialSearchMethod	LongWord	**OPTIONAL.** Get serial search method preference. Returns 0 if set to auto-scan all available ports. Returns 1..n if set to scan only one port (COM1..COMn). This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True.
SetSerialSearchMethod (*Value*)	nothing	**OPTIONAL.** Set serial search method preference to either auto-scan all available ports (default) or scan a single port only. This option allows a user to limit the scanning process to a specific serial port. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True. *Value* (LongWord) : 0 = Auto (scan all available ports), 1..n = Scan only one port (COM1..COMn).
GetSerialSearchRules	PChar	**OPTIONAL.** Get serial search rules preference as a zero-terminated string. This function need only be called if the application is handing the Propeller library's preferences manually (ie: InitPropellent was called with *StorePrefs* = False). The serial search rules may be changed by the user at any time that the ShowEditPorts routine is called (allowing them to review the rules and modify them as desired). This setting is

automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True. **NOTE: After calling this function, be sure to copy the returned string to the caller's memory space before executing other Propellent functions which return a PChar.**

SetSerialSearchRules (*Value*) — nothing

OPTIONAL. Call this to set the serial search rules preference if Propellent library is configured to not store its own preferences (ie: InitPropellent was called with *StorePrefs* = False). If Propellent library is configured to store its own preferences (recommended), simply ignore the Set/Get- SerialSearchRules routines. The format and content of the serial search rules preference is all handled by the Propellent library; do not edit the value, but rather, call this procedure if you wish to manually set it to a value saved from a previous session. To get the current setting call the GetSerialSearchRules function. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True.

Value (PChar) : A pointer to a zero-terminated string containing the serial search rules to use.

GetLibraryPath — PChar

OPTIONAL. Get the path to the Propeller Library files that will be used for source code compiling operations. This setting is automatically saved and restored between sessions by the Propellent library if InitPropellent was called with *StorePrefs* = True. **NOTE: After calling this function, be sure to copy the returned string to the caller's memory space before executing other Propellent functions which return a PChar.**

SetLibraryPath (*Value*) — nothing

REQUIRED IF USING COMPILE FEATURES. Call this to set the path to the Propeller Library files that will be used for source code compiling operations. The library path must be set before calling CompileSource. If library path was set in a prior session, and InitPropellent was called with *StorePrefs* = True, there is no need to call SetLibraryPath unless the path needs to be changed.

Value (PChar) : A pointer to a zero-terminated string containing the fully-qualified path to the Propeller library; example: C:\Program Files\Parallax Inc\Propeller Tool

ShowEditPorts — nothing

OPTIONAL. Call this to display the Serial Port Search List dialog. This dialog describes the computer's serial ports and the order in which the library will search them when scanning for a Propeller chip. The user is able to reorder the list of ports and can create simple or advanced rules indicating which serial ports to include or exclude for scanning. The library manages the settings created by the user via this dialog and, if InitPropellent was called with *StorePrefs* = True, automatically stores those settings in the Windows registry for persistance across sessions.

GetPorts (*Scannable*) — PChar

OPTIONAL. Call this to receive a pointer to a string containing a comma-delimited list of current serial ports on the computer. Call this at any time to get an updated list since the available serial ports can change at any time as USB-to-Serial devices are connected/disconnected from the system. **NOTE: After calling this function, be sure to copy the returned string to the caller's memory space before executing other Propellent functions which return a PChar.**

Scannable (Boolean) : True = (recommended) string is filtered per preferences to include only desirable ports. False = string is unfiltered; shows every serial port currently on the computer in the order given by the system.

CompileSource (*Filename*, *ShowError*) — Pointer

OPTIONAL. Call this to compile a Propeller application from source code. Returns 0 (null) if compile was successful; returns non-zero if an error occurred. After a successful compile, call DownloadToPropeller with a null *Filename* parameter to download this compiled Propeller application to the Propeller chip. NOTE: Propeller source files may be either ANSI-encoded (one byte per character) or Unicode-encoded (UTF-16; two bytes per character).

RETURN VALUE : 0 (null) = compile was successful.

Non-zero = compile failed. Returned value is a pointer to the following structure:

SrcFile (PChar)	:	A zero-terminated string containing the fully qualified path and file name of the source code that caused the error.
SrcStart (LongWord)	:	The first character of the source causing the error (0-based).
SrcCount (LongWord)	:	The number of characters causing the error.
Error (PChar)	:	A zero-terminated string containing the error text message.

Note that Propeller source files may be either ANSI-encoded (one byte per character) or Unicode-encoded (UTF-16; two bytes per character). Also, before compiling, the source is automatically translated to use single carriage return (CR) characters for line ends. Keep these details in mind when interpreting the SrcStart and SrcCount fields of the returned error structure.

Filename (PChar) : This parameter must either be null or must be a pointer to a zero-terminated string containing the fully-qualified path to a folder or to a file that is the top object (source code) to compile. The Propellent Library will automatically load and compile any subobjects called for in the *Filename* object as needed.

Filename <> null: If *Filename* indicates a path to a folder, an Open dialog will appear (with that folder selected) to allow the user to choose a source file to be compiled. NOTE: To specify a folder, *Filename* must end with a trailing path delimiter '\'; for example: C:\Temp\ indicates the Temp folder on the C drive. If *Filename* indicates a valid path and file, that file will be opened automatically (no dialog will appear) and will be compiled.

Filename = null: An open dialog will appear with the last-accessed folder selected to allow the user to choose a source file to be compiled.

ShowError (Boolean) : True = on error, show an error dialog before returning to caller. False = on error, immediately return to caller.

CompileSourceDocs (*Filename*) PChar **OPTIONAL.** Call this to compile and retrieve documentation for source file given by *Filename*. Returns 0 (null) if failed; returns pointer to zero-terminated document string if successful.

RETURN VALUE : 0 (null) = failed.
Non-zero = success. Returned value is a pointer to a z-string containing the document.

Filename (PChar) : This parameter must either be null or must be a pointer to a zero-terminated string containing the fully-qualified path to a folder or to a file to retrieve the document for.

Filename <> null: If *Filename* indicates a path to a folder, an Open dialog will appear (with that folder selected) to allow the user to choose a source file to retrieve the document for. NOTE: To specify a folder, *Filename* must end with a trailing path delimiter '\'; for example: C:\Temp\ indicates the Temp folder on the C drive. If *Filename* indicates a valid path and file, that file will be opened automatically (no dialog will appear).

Filename = null: An open dialog will appear with the last-accessed folder selected to allow the user to choose a source file to retrieve documentation from.

GetSourceHierarchy — PChar

OPTIONAL. Call this after a successful call to CompileSource to retrieve the structure of the Propeller Application just compiled. The returned value is a pointer to a contiguous array of elements representing the objects of the application from the first object (the top object) to the last object (the last one referenced). Each element of the array returned consists of the following:

1 byte	=	Indicates the attributes (type, location, horizontal level) of the object in the hierarchy.
	Bit 6	: 0 = source file, 1 = data file
	Bits 5:4	: 00 = unused, 01 = work folder, 10 = library folder, 11 = dual (exists in both work and library folders)
	Bits 3:0	: Horizontal level of object (0 through 15), where 0 is the left-most level (the top object) and 1 through 15 are deeper levels (nestings) of child objects.
n bytes	=	full path and file name as a zero-terminated string
1 byte	=	zero-terminator for the above string

The end of the array is denoted by one last element that consists of just 2 bytes, both equal to zero.

SaveImage (*AsBinary*, *Filename*, *ShowSaveAs*) — Boolean

OPTIONAL. Call this after a successful call to CompileSource to save the compiled Propeller Application as a binary or EEPROM image.

AsBinary (Boolean) : True = save image as a .binary file. False = save image as a .eeprom file.

Filename (PChar) : A pointer to a zero-terminated string indicating the fully qualified path and filename for the image file to create. This path and filename will appear in the Save As dialog, if *ShowSaveAs* is True. Path and filename may be the source's file name, in which case the extension will be replaced with either .binary or .eeprom accordingly. If *Filename* ends with a path delimiter, \, the image will be saved in that path with the original source filename and the appropriate extension. If *Filename* ends with a non-path delimiter, the image will be saved using that path and filename; any given extension will be replaced with .binary or .eeprom. Any folders in the path string that do not exist will be created automatically.

ShowSaveAs (Boolean) : True = show Save As dialog (containing *Filename's* path and file) for user to make adjustments as desired. False = do not show Save As dialog; *Filename* must be a valid path and filename in this case.

GetPropellerVersion — PChar

OPTIONAL. Scan serial ports for a Propeller chip and display and return the port and version of the Propeller chip (if found) or an error message (if not found). **NOTE: After calling this function, be sure to copy the returned string to the caller's memory space before executing other Propellent functions which return a PChar.**

RETURN VALUE : Returned value is a pointer to a z-string containing one of the following:
if success: `COM# : VERSION #`
if failure: <error message>

DownloadToPropeller (*Filename*, *DownloadCmd*) — nothing

OPTIONAL. Scan serial ports for a Propeller chip and download a compiled Propeller Application to it. The "application" may be either an existing image file or the last successfully compiled application.

Filename (PChar) : This parameter must either be null or must be a pointer to a zero-terminated string containing the fully-qualified path to a folder or file.

		Filename <> null: If *Filename* indicates a path to a folder, an Open dialog will appear (with that folder selected) to allow the user to choose an image file to be downloaded. NOTE: To specify a folder, *Filename* must end with a trailing path delimiter '\'; for example: C:\Temp\ indicates the Temp folder on the C drive. If *Filename* indicates a valid path and file, that file will be opened automatically (no dialog will appear) and will be downloaded to the Propeller. *Filename* = null: The image of the last successfully compiled application (via a call to CompileSource) will be downloaded to the Propeller.
	DownloadCmd (Byte) :	1 = write to RAM and run, 2 = write to EEPROM and stop, 3 = write to EEPROM and run. NOTE: command 2 is not currently used by the Propeller Tool software.
FinalizePropellent	nothing	**REQUIRED.** This routine cleans up internal items, like memory, used by the library. This must be called just before unloading the Propellent Library (if dynamically linked), or just before application termination (if statically linked).