How to write your own object
By Microcontrolled

If you are a Propeller user of any skill level, you may have come to the point where there is not an object available for something you want to do, and you want to venture out and make your own. Or maybe you just want to expand the Propeller Object Library. Ether way, this guide will tell you how to make your own object and submit it to the Propeller OBEX.

Step 1: Check for copies

The last thing you would want to know when writing an object is that someone else has made exactly the same thing. Check the Propeller OBEX (found though the Parallax site) to make sure no one has already made it. Also check the Parallax Forums, and maybe even post a thread asking if it has been done. If it has, you can still write your own object, but it should be better then the current one or approach the job at a different angle.

Step 2: Get Opinions

Go online, post a thread in the Parallax forums about your object, and get the users opinions. The opinions you get will reflect what the majority of the people want, so listen carefully. If you don't think you can do the things they suggest, tell them before you start writing the object and get a second opinion.

Step 3: Start Coding!

This is the fun part, now you get to write the code for your object. Writing an object is a little different then writing a program, so here are some pointers to follow:
1: Include a Start function with pin settings (if applicable) and other constants. Some people name the function you call first “Init” or “Initialize”, but “Start” is basic and easy to remember if you are using multiple objects.
2: Name your variables and constants according to their purpose. This makes your code easier to follow so that others can modify it to suit their purposes if they wish to. Good examples include “relay_state” and “Input_flag”. Bad examples include “Var1” or “hol”. Also, the standard variable name for the incrementing variable in a repeat...from loop is “i”, and since repeat...from loops may be used in many different routines, “i” should be declared as a local variable and not a global variable.
3: If you include routines that you use in your object but are not intended for use by the user, label them as PRI routines instead to avoid confusion. Also, since PRI routines don't show up in Documentation mode, don't put any double quote comments in them.
4: Comment your code. You should have a comment section at the top of every PUB routine explaining how to implement the object into code. They should be double single-quotes “ '' “ or double brackets “ {{“ so that this will show up in Documentation mode. Comments throughout are appreciated, but too many make your code hard to work with.
5: Write a useful description for your code. This should be at the top of the code in double brackets, and include the name of the program, the name of the author, the version number (initial release is v.1.0), the date it was made/released, and the hardware required to operate. You should also have a short description explaining how the program works and how it should be used.

Step 4: Build a test program

It is a good idea to build a test program not only to test if your object functions as promised, but also to provide the user with an easy way to see that he has his hardware setup right and his pin settings correct. A test program should test every/almost every function offered by your object, and should use the Parallax Serial Terminal as feedback with the exception that the object is for a display itself. Several users use the TV_Text object or VGA_Text for feedback, and some even a serial LCD, but the one thing that is not hardware dependent is the Serial Terminal, so you should stick with that. If you need graphics support, PropTerm is a good option. By standard you should use a baud rate of 9600, unless you absolutely *need* faster data speed, and you should use the *unaltered* FullDuplexSerial object. Before starting the terminal engine, you should include the line "waitcnt(clkfreq*5 + cnt)", to give the user enough time to get the terminal up after loading the program into RAM.

Step 5: Check and Check

Check though your code. Make sure there are no errors in the coding, no misspelling in the comments, etc. Most importantly, make sure the object works as promised. Look though the code for things that can optimized or removed for memory or speed concerns. You and your users will want a well oiled object, and if it has one thing they dislike, they will probably not continue to use it.

Step 6: License

At the bottom of your code, put a DAT section there (if there is not one there already), and put the MIT licensing agreement in the bottom of the code. It can be found in the "License" object included with every copy of the Propeller Tool v.1.2.5 and greater and in every Parallax Propeller object included with the IDE.

Step 7: Acknowledge

Does your object use other objects to operate? If so, it is a courtesy to acknowledge the authors of the other objects in your description. If they followed this guide, their name should be at the top of the program in the description. If not, you can track them down on the OBEX or on the forum. A username will do, and they will appreciate it.

Step 8: Publish!

Once you've checked though everything and make sure everything is in working order, then go to the Propeller OBEX (http://obex.parallax.com/) and register/sign in. Then at the top of the page and select "Upload an Object" and fill out the form given. In a few moments after submitting it, your program will show up in the proper category.

Congratulations! You just created your own object!