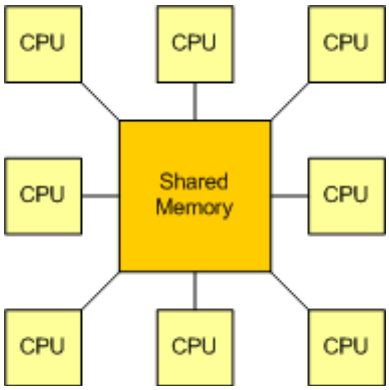
Multiprocessing Using Multiple Cogs:



Until this point we have been writing programs which are linear. They start, they continue along, and then unless they loop endlessly, they stop. Now we're going to make a big leap, the equivalent of running several programs at the same time.

One big feature of the Propeller is that it has 8 independent processors (called cogs). Each cog can be executing a different function. Functions can share data with functions in other cogs by using global variables which are stored in shared memory in what is called the hub.

If you don't remember the difference between multitasking and multiprocessing, go back and review the information at http://learn.parallax.com/propeller-brains-your-inventions/multitasking and http://learn.parallax.com/propeller-brains-your-inventions/multiprocessing.

Sometimes you need to control multiple outputs having different timing restraints. It might be possible to do it with a single processor, but can be extremely difficult. Multiple processors can make this kind of program much easier. It is infinitely easier to program complex behaviors into a robot using multiple processors.

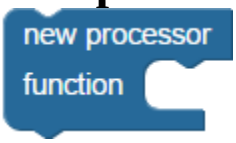
The way to use multiple cogs to solve a monster problem is to break the problem down into several (self-contained) functions and then send each to a different cog to be executed. The individual functions can pass values back and forth to other cogs or the main program using global variables (in shared memory).

Now go on to http://learn.parallax.com/multicore-approaches and follow the links at the bottom of the page to become familiar with using more than one cog. At the end of this you should be able to write functions, run them in separate cogs, share data using global variables, and stop a cog when its job is finished.

If you are wondering how you can use multiple cogs, the answer is fairly easy.

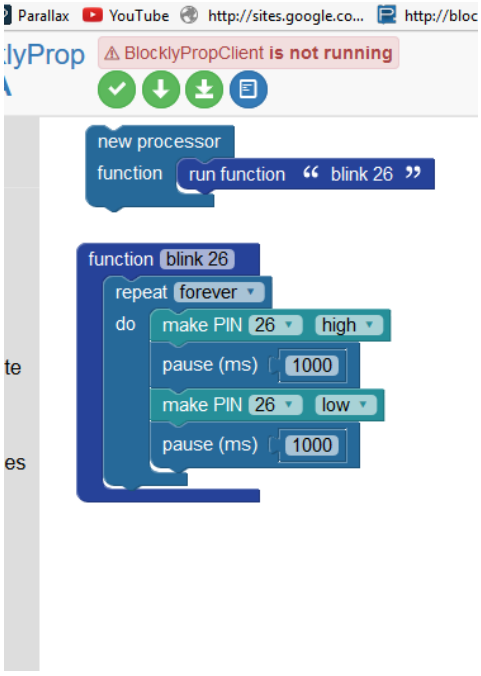
1. Choose a sequence of operations which you want the cog to perform
2. Write a function to include those operations
3. Use global variables to pass values to or from the function.
4. Use the following BlocklyProp information from Parallax:

5. new processor

6. 

7. The **new processor** block launches a function into its own processor (cog). There are up to 7 processors available on the Propeller. Some other blocks, especially blocks for devices, also use processors, limiting the total number of new processors that can be launched using this block. For example, if you are using 3 processors in one area then only 4 remain for use. Only a single **run function** block can be placed inside this block. Do not use functions that contain Terminal blocks; Terminal blocks and functions containing Terminal blocks can only run from the main program.

So, if you want to blink the LED on pin 26 while other stuff is continuing to go on the program fragment would look like this



The function is named blink26. It is called from the main program using the New Processor Function block.