# [**Module 3: Propeller C – Functions**](http://learn.parallax.com/propeller-c-functions)

**Alternate Text:** [**Procedures and Functions**](https://en.wikibooks.org/wiki/C_Programming/Procedures_and_functions)

This is a more in depth look at what functions are. Writing a C Program is really nothing more than racking-and-stacking small chunks of code in the right order and having a compiler stitch them together.

In this section you will learn about how to write and install your own functions in a program to make them less repetitious and more efficient, how to send the function information, and then get some back.

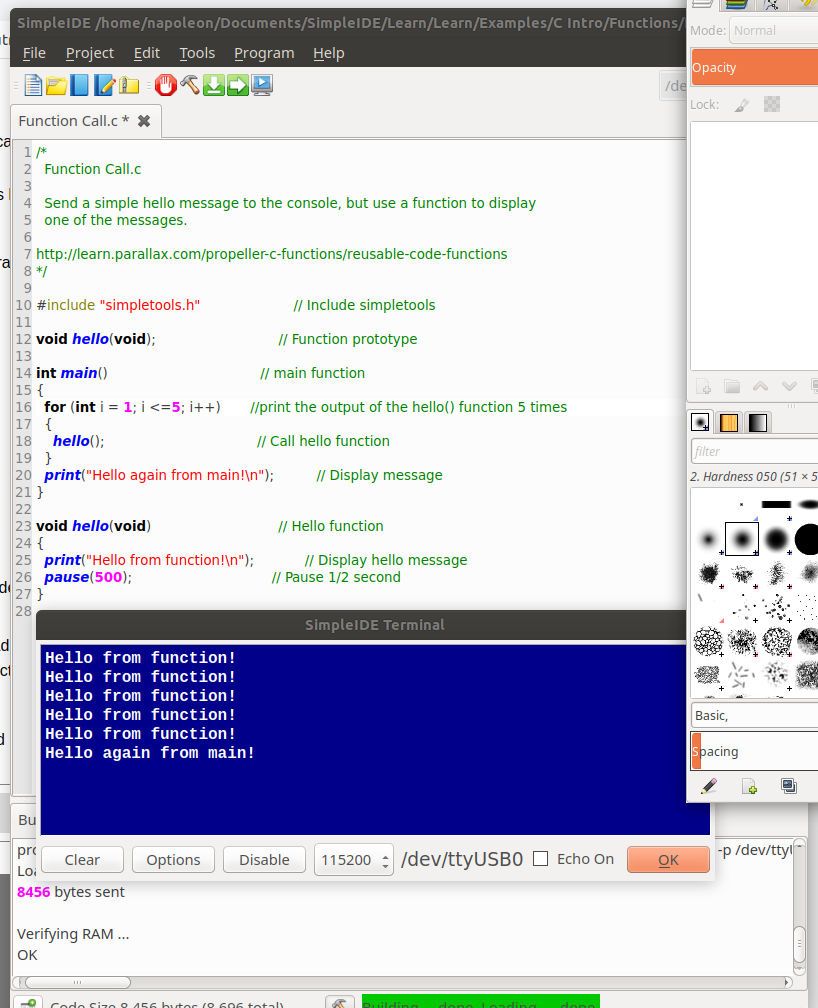
You will create “global variables” so that other parts of your program can have access to that data.

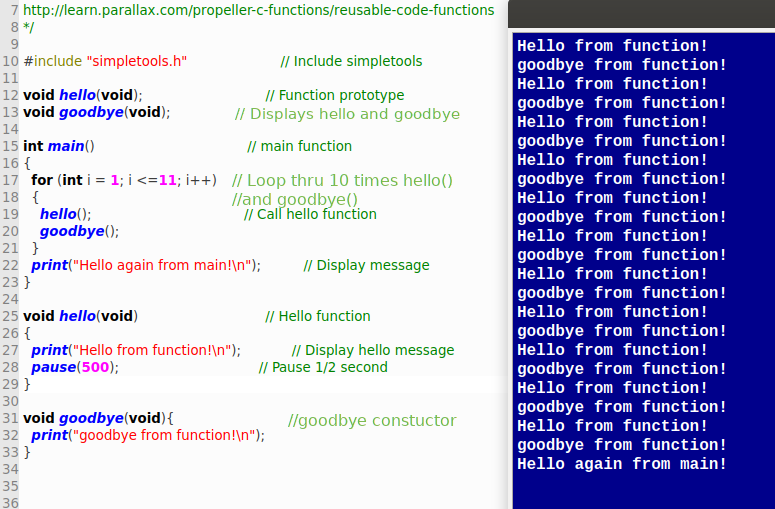
And finally, the last section demonstrates a key feature of the Propeller microcontroller. This chip has not one but eight separate computers on-board. You’ll see how other “cores” (also called “cogs”) can be started, run a program, stop and shut down.

[Reusable Code Functions](http://learn.parallax.com/propeller-c-functions/reusable-code-functions)

Why rewrite a section of code every time you want to run it. Why not write it once and call it up in your program whenever you need it. This section shows you how to write code that is used repetitiously in a program and tell compiler about it.

Insert a picture here and paste in a copy of your program for **Try This** for **Reusable Code Functions**. Don’t forget to adjust the comments for the program.

 Insert a picture here and paste in a copy of your program for **Your Turn** for **Reusable Code Functions**. Don’t forget to adjust the comments for the program.

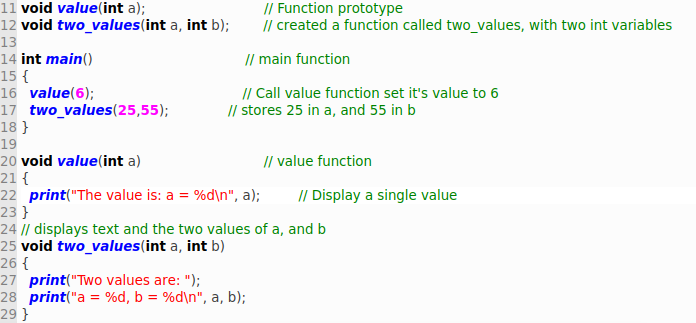


[Function With Parameter](http://learn.parallax.com/propeller-c-functions/function-parameter)

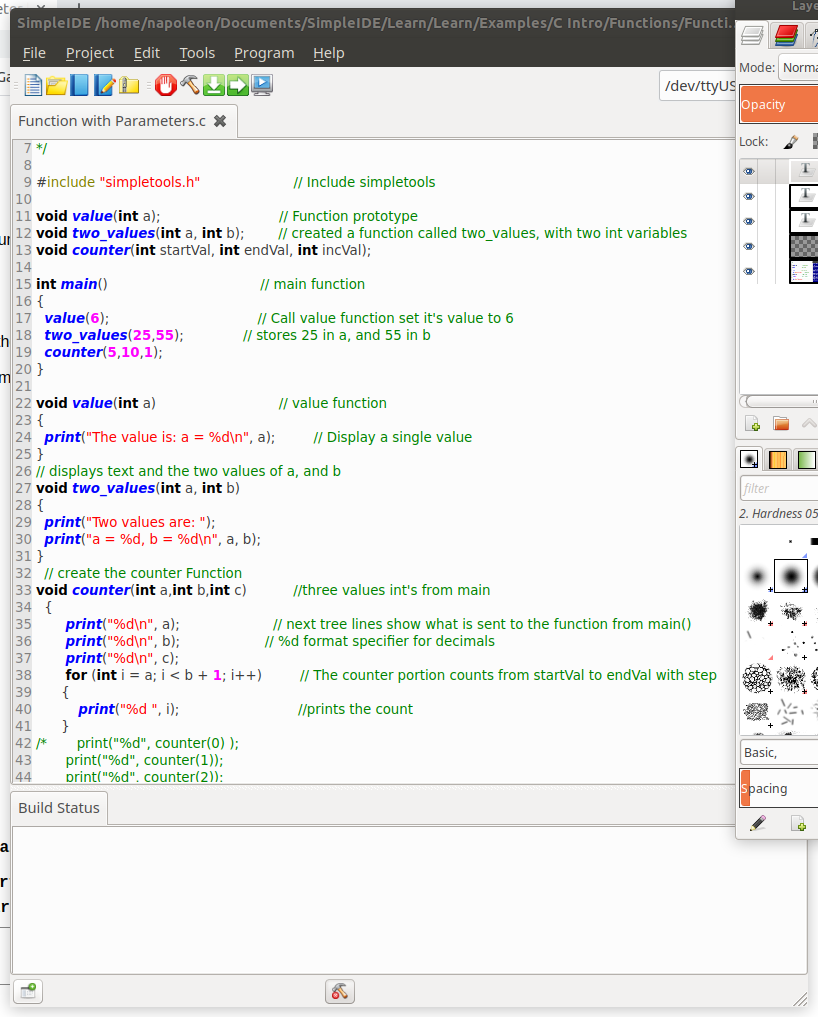
Even a simple function like **pause ( )** requires information from you. Pause? Ok… How long? In milliseconds if you please.

You’ve just passed a parameter to pause. As you might guess, it doesn’t need to be a number. It can be the name of a variable that holds the number. Learn how functions that require a parameter are crafted and added to a program.

Insert a picture here and paste in a copy of your program for **Try This** for **Function With Parameter**. Don’t forget to adjust the comments for the program.



Insert a picture here and paste in a copy of your program for **Your Turn** for **Reusable Code Functions**. Don’t forget to adjust the comments for the program.

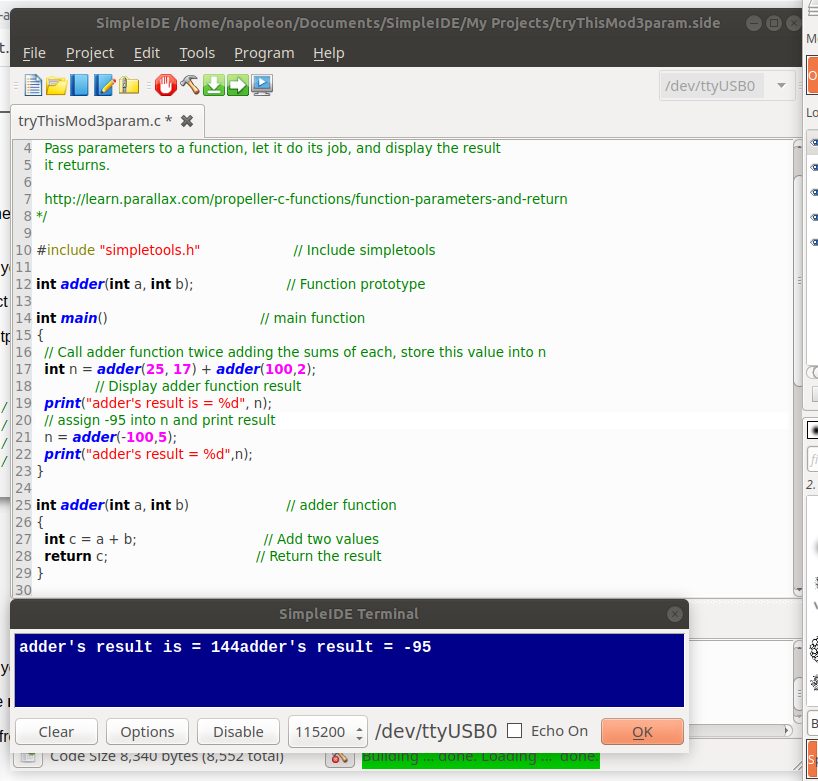


[Function With Parameters and Return](http://learn.parallax.com/propeller-c-functions/function-parameters-and-return)

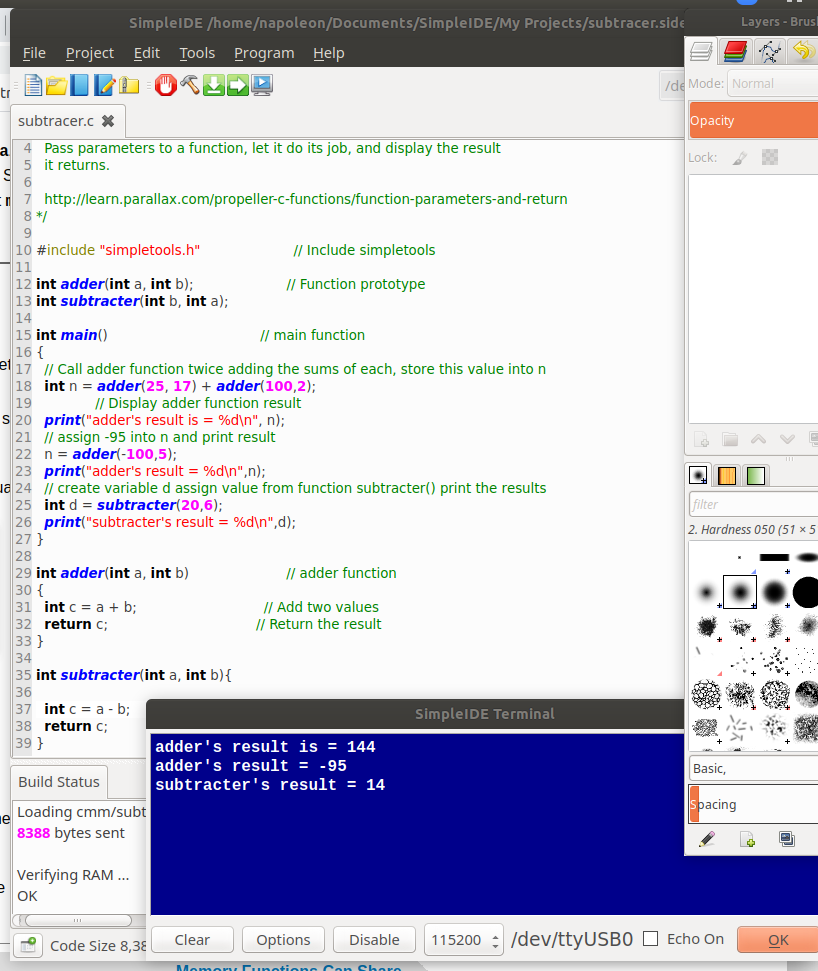
If you write a fancy math function and pass it all the necessary numbers, I would imagine you would want an answer from it. All functions should return to the program that called it. No function should provide a separate “way out”. That’s a rule. This “loose style” of programming is known as “spaghetti code”. It’s difficult to troubleshoot and debug.

Many of this type of function may create some of its own variables called **local variables**. When the function finally returns to the main program, any memory space occupied by those variables is released.

Insert a picture here and paste in a copy of your program for **Try This** for **Function with Parameters and Return**. Don’t forget to adjust the comments for the program.



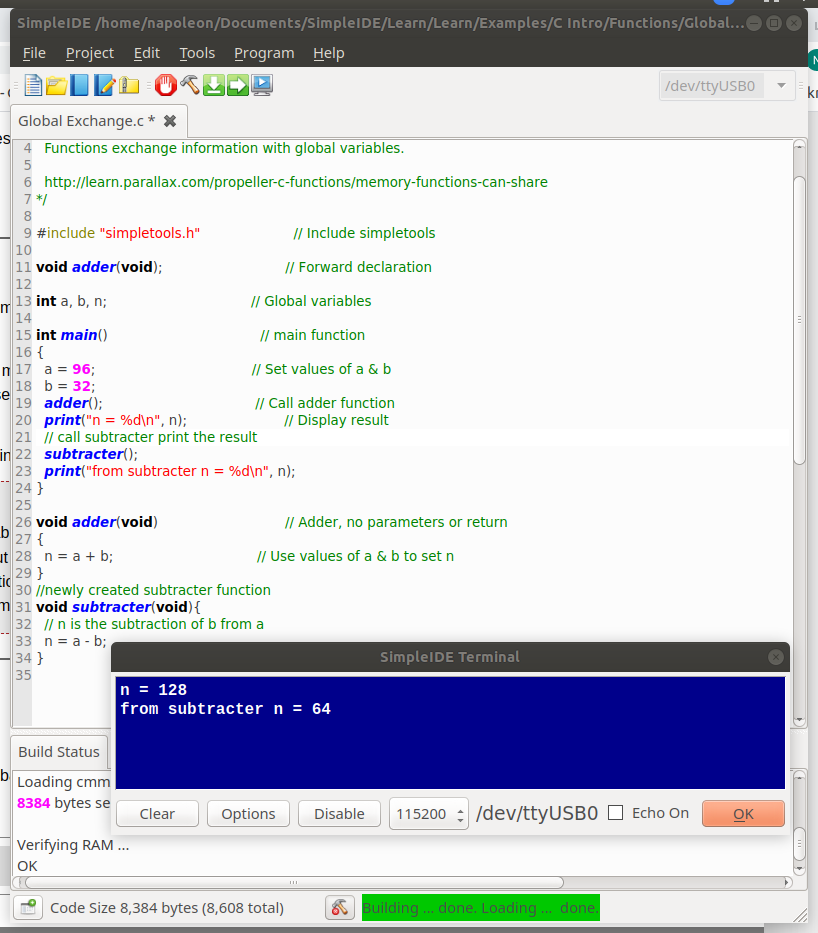
Insert a picture here and paste in a copy of your program for **Your Turn** for **Function with Parameters and Return**. Don’t forget to adjust the comments for the program.



[Memory Functions Can Share](http://learn.parallax.com/propeller-c-functions/memory-functions-can-share)

Your functions can use **global variables** and return to global variables. Instead of just returning the parameters back to the calling program, it can simply deposit the answers in commonly accessed memory as global variables. That way any part of your program can make use of those values provided by the function. This is just another way to pass numbers around…

Insert a picture here and paste in a copy of your program for **Try This** for **Memory That Functions Can Share**. Don’t forget to adjust the comments for the program.



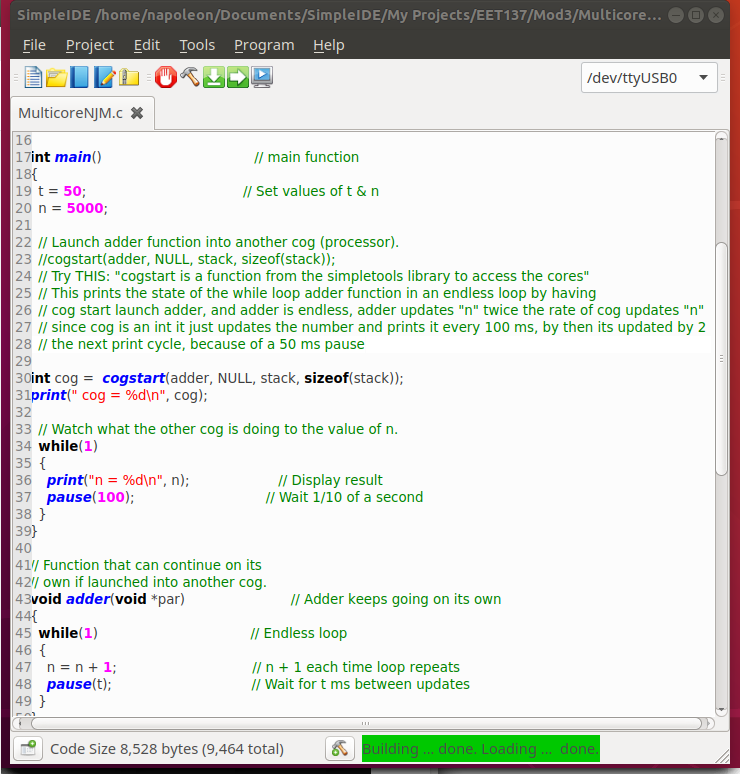
[Multiple Core Use: An Example](http://learn.parallax.com/propeller-c-functions/multicore-example)

Yes, the Prop is a unique little bug…. It has lots of processing power in there. If you read through the “What’s a Microcontroller?” at the top of the Parallax tutorials you might realize that.

For example, in most systems, hardware that might need processing time accesses it through an elaborate technique known as **interrupt handling**. The central processor will drop what it’s doing and start another program that “handles” the specific hardware. When a return is detected in that program, the processor returns to what it was doing at the time the interrupt was serviced. Getting all this worked out in a program, for priority (what gets service first), and timing, and sequencing is a large part of system programming.

The Propeller doesn’t do that. It simply starts up another computer and hands the job over to it. When the job is done, the computer is shut off waiting for the next “customer”. This section gives you a little insight into a different bend on handling interrupts. Very little is done by the “main cog”.

Insert a picture here and paste in a copy of your program for **Try This** for **Multiple Core Use**. Don’t forget to adjust the comments for the program.



Insert a picture here and paste in a copy of your program for **Your Turn** for **Multiple Core Use**. Don’t forget to adjust the comments for the program.

