C language

Collection Editor:
Thiyagarajan S

C language

Collection Editor:
Thiyagarajan S

Authors:
Kileen Cheng
David Waldo

Online:
< http://cnx.org/content/col11596/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: December 14, 2013
PDF generated: December 14, 2013
For copyright and attribution information for the modules contained in this collection, see p. 7.

Table of Contents

Chapter 1

C Programming Basics[1]

[1] This content is available online at <http://cnx.org/content/m11853/1.1/>.

Chapter 2

C Programming Lab[1]

2.1 Skills

This laboratory deals with compiling and running a C program on the TMS320C6713 DSP using Code Composer Studio. By the end of the laboratory you will be able to:

- Compile and build a C language program
- Run a C language program on the TMS320C6713 DSP
- Link multiple files with functions into one project

2.2 Reading

- SPRU 187: Optimizing C Compiler
- C programming reference card[2]

2.3 Description

If you have forgotten how to program in C/C++ you should review C programming on the many tutorials on the internet or look through your favorite textbook or reference book. This document will not give you a tutorial on how to program in C/C++.

You must remember that when programming the TI DSP you are loading your program into a system that has its own DSP and memory and is apart from your computer. In order for you to print anything, the data must be sent from the DSP board to CCS and then displayed in the stdio display. This can be very slow so you will want to use the debugging tools to view variables.

In this lab you will be writing very simple C programs. If you want to print something you will obviously need to include the stdio.h file.

Your main program should look like:

```
#include <stdio.h>
main()
{
variables
code
}
```

[1] This content is available online at <http://cnx.org/content/m36345/1.3/>.
[2] http://www.cise.ufl.edu/class/cop4600/docs/C_Ref_Card.pdf

Be sure to use good programming style.

2.4 Pre-Laboratory

Write a C program that performs a dot product of the following vectors. The data for `a` and `b` should be stored in integer arrays. Use a *for* loop.

a = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30]
b = [30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1]

$$y = \sum_k a_k \cdot b_k \tag{2.1}$$

2.5 Laboratory

2.5.1 Part 1

- In this part you will create a project that performs the array multiply and sum within the main function of the program. In order to be able to accurately count clock cycles, use the C6713 simulator.
- Create a new project, call it clab.
- Create your C file. Call it `main.c`. Add this file to your project.
- You should use the following options for the project (Project->Build Options):
- After the project has been loaded, record the values in the variables at each step as you single step through the program. Count the number of clock cycles and compare to the assembly program completed in a previous lab. Be sure to zero the counter before stepping through your code.

2.5.2 Part 2

- In this part you are going to write a function that performs an array multiply and sum algorithm and returns the sum. This function will be located in another file and you will also make a header file.
- Create a C file called `multsum.c`. In this file make a function called `multsum`. The function should have as inputs:

- pointer to integer array `a`
- pointer to integer array `b`
- number of elements in the arrays
- The output of the function should be the integer sum. The function should perform the multiply and sum algorithm.

- Create a header file for the multsum function and call it `multsum.h`. In the header include the following code:

```
  #ifndef _MULTSUM_H_
#define _MULTSUM_H_
extern int multsum(int *a, int *b, int length);
#endif /*MULTSUM_H_*/
```

This uses some compiler directives to determine if the current header file has already been included by another include file. The macro `_MULTSUM_H_` is not defined the first time it is encountered so the macro is defined and the definition is included. The `extern` keyword tells the compiler that the function is defined in another file.

- In `main.c`:

- include the `multsum.h` file

```
#include "multsum.h"
```

- declare the `a` and `b` arrays
- call the `multsum` function with the sum returned to a variable

- Include the `multsum.c` file in your project.
- Run the program and verify that the sum that is returned is correct.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *C language*
Edited by: Thiyagarajan S
URL: http://cnx.org/content/col11596/1.1/
License: http://creativecommons.org/licenses/by/3.0/

Module: "C Programming Basics"
By: Kileen Cheng
URL: http://cnx.org/content/m11853/1.1/
Page: 1
Copyright: Kileen Cheng
License: http://creativecommons.org/licenses/by/1.0

Module: "C Programming Lab"
By: David Waldo
URL: http://cnx.org/content/m36345/1.3/
Pages: 3-5
Copyright: David Waldo
License: http://creativecommons.org/licenses/by/3.0/

C language
This website "http://fresh2refresh.com/c/c-language-history/" will be very useful for students who are beginners for C programming. * All topics are explained very clearly and in very easy way to understand. * Easy navigation through all topics * Simple example programs and output * Real time application programs with source code

About Connexions
Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.