CPE 325: Embedded Systems Laboratory Laboratory #1 Tutorial Introduction to TI Code Composer Studio (IDE)

Aleksandar Milenković

Email: <u>milenka@uah.edu</u> Web: <u>http://www.ece.uah.edu/~milenka</u>

Objective:

This tutorial will help you get started with the TI's Code Composer Studio for MSP430. It includes the following topics:

Creating an application project Debugging

Notes:

The latest version of Code Composer Studio can be downloaded for free from the TI's web site: <u>http://www.ti.com/tool/ccstudio.</u>

Contents:

1	Cre	ating a Project	2
	1.1	Creating a New Workspace and a Project	2
	1.2	Compiling and Linking the Application	
2	Deb	pugging	
	2.1	Starting the Debugger	10
	2.2	Inspecting Source Statements	11
	2.3	Inspecting Variables	12
	2.3.	1 Setting a Watchpoint	12
	2.3.	2 Setting and Monitoring Breakpoints	. 14
	2.3.	3 Executing up to a Breakpoint	14
	2.4	Debugging in Disassembly Mode	14
	2.5	Monitoring Registers	15
	2.6	Monitoring Memory	16
	2.7	Viewing Terminal I/O	18
3	Soft	tware Documentation	18
	3.1	Code Formatting and Organization	
	3.2	Code Headers and Comments	19
	3.3	Software Flowcharts	. 19

1 Creating a Project

This section introduces you to TI's Code Composer Studio (CCStudio) Integrated Development Environment (IDE) that will be used for software development in the Embedded Systems Laboratory. In the form of a step-by-step tutorial, it demonstrates a typical development cycle and shows you how to use the CCStudio compiler and the CCStudio linker to create a small application for the MSP430 microcontroller. It includes topics such as creating a workspace, setting up a project with C source files, compiling and linking your application, and debugging.

1.1 Creating a New Workspace and a Project

Using the Code Composer Studio IDE, you can design advanced projects. For more information, read the Code Composer Studio wiki at <u>http://processors.wiki.ti.com/index.php/Category:CCS.</u> Here, we will create a relatively simple project with several source files.

Step 1. Open Code Composer Studio (CCS).

Step 2. In the Eclipse Launcher select a directory to be used as the workspace (see Figure 1.) and press <Launch>. A *Getting Started* window will appear (Figure 2). This page provides links to example projects as well as documentation related to Code Composer and TI Microcontrollers.

👽 Eclipse Launcher	×
Select a directory as workspace	
Code Composer Studio uses the workspace directory to store	its preferences and development artifacts.
Workspace: C:\Users\milenka\workspace_cpe325	✓ Browse
workspace: C.(Users/Initerika/Workspace_cpescs	browse
Use this as the default and do not ask again	

Figure 1. Eclipse Launcher

Step 3. Create a new project from the *Getting Started* menu. Click the <New Project> button. A New CCS Project window appears as shown in Figure 3.



Step 4. Name the project "Lab1_D1" and select "Empty Project" (see Figure 3.). For the target, select one of the following options depending on the target board: (a) MSP430FG4618 for the TI MSP430FG4618/2013 Experimenter's board; or (b) MSP430G2553 for the MSP430 Launchpad.

Step 5. Click <Finish>. The Project Explorer will show where all the project files are. Note that there are no source code files because we created an Empty Project (Figure 4).



Figure 3. New CCS Project Setup



Figure 4 Project Explorer

Step 6. Copy the "Lab1_D1.c" and "twofact.c" from Windows Explorer to Lab_D1 in the project explorer. Right click on Lab1_D1 and select Add Files option. When prompted select "Copy Files" and click <OK>.

😯 File Operation		×
Select how files should be imp	ported into the project:	
• Copy files		
O Link to files		
Create link locations	relative to: PROJECT_LOC	~
Configure Drag and Drop Setti	ings	
?	ОК	Cancel

Figure 5. File Operation Dialog

Step 7. Double-click one of the files to see the source code (see Figure 6).



Figure 6. Code Composer Studio

1.2 Compiling and Linking the Application

You can now compile and link the application. You should also create a compiler list file and a linker map file and view both.

Step 1. Right-Click "Lab_D1" in the project explorer and select "Properties". A new window will pop up as shown in Figure 7.

ype filter text	Build		<-> < <-> <
Resource General			
Build MSP430 Compiler Processor Options	Configuration: Debug [Active]		✓ Manage Configurations
Optimization Include Options ULP Advisor	Builder 🎽 Validator 斗 Steps 5	👌 Variables 🛛 🌄 Environment 😽 Link Orc	ler 🛱 Dependencies
Advice Options	Use default build command		
Predefined Symbols > Advanced Options	Build <u>c</u> ommand: \${CCS_UTILS_DIR}/bin	n/gmake -k -j 8	Variables
 MSP430 Linker MSP430 Hex Utility [Disabled] Debug 	Makefile generation		
	Build location		
	Build directory: \${workspace_loc:/Lab1	_D1}/Debug	Workspace File system Variables
	Build settings	Inable parallel build	 Use optimal jobs (8) Use parallel jobs: 8 + Use unlimited jobs
	Make build targets		
	Build on resource save (Auto build)	all	Variables
	Build (Incremental build)	all	Variables
	☑ Clean	clean	Variables
	See 'General' for changing tool versions an	d device settings	
Show advanced settings			Apply and Close Cancel

Figure 7. Project Properties

Step 2. From this menu, you can setup various project settings such as compiler optimizations, lst file generation, heap size, stack size, and the level of printf/scanf support. Below are several useful options.

- Select the "Build->MSP430 Compiler->Optimization." This option allows you to set optimization settings. Change the "Optimization level" to off.
- Select the "Build->MSP430 Compiler->Advanced Options->Assembler Options" to setup the listing file options. Select the checkbox "Generate listing file (--asm_listing, -al)" (Figure 8).
- Select the "Build->MSP430 Compiler->Processor Options." This allows you to set the code and memory model. Set the "Silicon version" to msp and the "code memory model" and "data memory model" to small. This will limit the instruction set to non-extended instructions.

✓ Build	Configuration: Debug [Active]	✓ Manage Configurations
 MSP430 Compiler 		
Processor Options		
Optimization	Keep the generated assembly language (.asm) file (keep_asm, -k)	
Include Options		
ULP Advisor	Generate listing file (asm_listing, -al)	
Advice Options	Source interlist	~
Predefined Symbols		
 Advanced Options 	Pre-define assembly symbol NAME (asm_define, -ad)	🗐 🔊 🙆 준티 준티
Advanced Debug Options		
Language Options		
Parser Preprocessing Opti		
Diagnostic Options		
Runtime Model Options		
Advanced Optimizations		
Entry/Exit Hook Options		
Feedback and Analysis Op		
Library Function Assumpt		
Assembler Options		
File Type Specifier		

Figure 8 Project Assembler Options

• Select the "Build->MSP430 Compiler->Advanced Options->Language Options". Change "Level of printf/scanf support requires (--printf_support) to full.

uild MCD 420 Committee	Configuration: Debug [Active]	✓ Manage Configuratio	ns.
MSP430 Compiler Processor Options			
Optimization Include Options	C Dialect	Compile program in ANSI C89 mode. (c89)	
ULP Advisor Advice Options	C++ Dialect	Compile program in C++03 mode. (c++03)	
Predefined Symbols	Language mode	Relaxed parsing (non-strict ANSI) (relaxed_ansi, -pr)	
 Advanced Options 	Enable C++ exception handling (exceptions)		
Advanced Debug Options	Support C++ run-time type information (rtti, -rtti)		
Language Options	Treat C files as C++ files (cpp default, -fg)		
Parser Preprocessing Opti Diagnostic Options	Allow extern C functions to propagate exceptions (extern_c_can_throw)		
Runtime Model Options	Floating point precision accepted by compiler (float_operations_allowed)	all	
Advanced Optimizations Entry/Exit Hook Options	Specify how to treat plain chars (signed/unsigned) (plain_char)	unsigned	
Feedback and Analysis Or	limit pending template instantiations (pending_instantiations)		
Library Function Assumpt Assembler Options	Level of printf/scanf support required (printf_support)	full	

• Select the "Build->MSP430 Linker->Basic Options" to set the heap size. Change "Heap size for C/C++ dynamic memory allocation (--heap_size, -heap) from 80 to 300. This change is necessary to get the printf debugging to work later in the lab. Click <Apply and Close> to close the properties windows.

✓ Build	Configuration: Debug [Active]	✓ Manage Configurations
> MSP430 Compiler		
 MSP430 Linker 		
Basic Options		
File Search Path	Link information (map) listed into <file> (map_file, -m)</file>	"\${ProjName}.map"
 Advanced Options 	Specify output file name (output file, -o)	"\${ProjName}.out"
Command File Preproces:	specify output me name (output_me, -o)	striojivame,out
Diagnostics	Heap size for C/C++ dynamic memory allocation (heap_size, -heap)	300
Linker Output	Set C system stack size (stack_size, -stack)	80
Symbol Management	Set C system stack size (stack_size, -stack)	80
Runtime Environment	Hold watchdog timer during cinit auto-initialization (cinit_hold_wdt)	on 🗸
Miscellaneous		
MSP430 Hex Utility [Disabled]		

Figure 10 Project Linker Options

Step 3. To build a single file right-click the file in the Project Explorer and select "Build Selected File(s)" as shown in Figure 11. The Console window will show you if there are any build errors (Figure 12).

	🍋 Project Explo	orer 🛛	3				
	✓ ≝ Lab_D1		ve - Debug]				
	> 🗊 Inclu						
	👝 Debu	-					
	> 👝 targe > 💽 Lab1		igs		_		
	> 📝 Labr		New	>			
	> 💽 twof		Open				
			Open With	>			
			Show in Local Terminal	>			
			Сору	Ctrl+C			
			Paste	Ctrl+V			
		*	Delete	Delete			
			Source	>			
			Move	,			
			Rename	F2			
		-	Import	>			
			Export		8		
			Show Build Settings				
1 3			Build Selected File(s)				
			Clean Selected File(s)			221	
	Figu		Exclude from Build 11. File Options	Menu	0	7.0	2
workspace_cpe325 - Lab1_D1/Debug/Lab1_	Y	ure		Menu	0		1
ile Edit View Navigate Project Run Scripts	_D1.map - Code Composer St Window Help	ure		Menu	0		
File Edit View Navigate Project Run Scripts	_D1.map - Code Composer St Window Help	ure				Quick Access	
File Edit View Navigate Project Run Scripts • • • • • • • • • • • • • • • • • • •	_D1.map - Code Composer St Window Help 巻 ▼ @ ▼ ♡ ◇ ▼ < ♡	ure Itudio	11. File Options	8		Quick Access	
Ile Edit View Navigate Project Run Scripts	_D1.map - Code Composer Si Window Help ╈ + @ + © ♀ + ⊄ printr_Supporterui	itudio	11. File Options	" \$ptay_error_numoer	asm_iisting -2 -m	Quick Access	
File Edit View Navigate Project Run Scripts The Edit View Navigate Project Run Scripts Console & Console	_Dl.map - Code Composer Si Window Help ≱ + @ • !© ♀ < ¢ printr_support=tuil 80cinit_hold_wdt= /ti-gt-msp430_18.1.;	tudio	11. File Options	Spiay_error_numeer " "ticget-msp430_18.1.1.	asm_iisting -2 -m LTS/include"rer	Quick Access	
<pre>iile Edit View Navigate Project Run Scripts Edit View Navigate Project Run Scripts Console # Console # Of Build Console [Labl.01] beap.size=390stack.size -1°c:/ti/ccsv8/tools/compiler diag.wrap=offdisplay_err "./Labl_D1.obj" ./twofact.ob (Linking></pre>	_Dl.map - Code Composer Si Window Help * * @ • @ • @ • @ printr_supporterui 88cinit_hold_wdt+ /ti-cgt-msp430_181 /ti-warn_sec _j" "/lnk_msp438fg44	ture	11. File Options	Spiay_error_numeer " "ticget-msp430_18.1.1.	asm_iisting -2 -m LTS/include"rer	Quick Access	
<pre>iile Edit View Navigate Project Run Scripts</pre>	D1.map - Code Composer St Window Help ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	tudio	11. File Options	Spidy_error_numeer	asm_iisting -z -m LTS/include"rer -rom_model -o "Lab	Quick Access	
le Edit View Navigate Project Run Scripts	D1.map - Code Composer St Window Help + • @ • • • • • • • • • • • • • printr_support=tuit 80cinit_hold_wdt= printr_support=tuit 80cinit_hold_wdt= j*/hk_msp336f4 ttected no uses of lo ttected uninitialized	itudio itudio	11. File Options	spidy_error_numeer " "ticgt-msp43@_18.1.1. ml"use_hw_mpy=16 - lizing all unused por	asm_IISLINg -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa	Quick Access	
e Edit View Navigate Project Run Scripts • Image: State Project Run Scripts • Image: State Project Run Scripts • Console Image: State Project Run Scripts • Consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins.	D1.map - Code Composer St Window Help + +	itudio itudio	11. File Options	Spidy_error_numuer " "t-cgt-msp43@_18.1.1. ml"use_hw_mpy=16 - lizing all unused por lizing all unused por	asm_listing -2 -m LTS/include"rer -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
e Edit View Navigate Project Run Scripts • Image: State Project Run Scripts • Image: State Project Run Scripts • Console Image: State Project Run Scripts • Consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins. • remark #10372-D: (ULP 4.1) De consumption on unused pins.	_D1.map - Code Composer Si Window Help # → @ → [@ → [© ۞ → ⊂ printr_Supporterui. +80cinit_hold_wdt+ -/ti-cgt-msp430_18.1. ti-tamsp430_18.1. +ti-cgt-msp430_18.1.	Aureau Autoria	11. File Options	Spidy_error_numuer " ti-cgt-mspd3@_18.1.1 ml"use_hw_mpy=16 - lizing all unused por lizing all unused por lizing all unused por	asm_IISLINg -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa ts to eliminate wa	Quick Access	
<pre>e Edit View Navigate Project Run Scripts</pre>	_Dl.map - Code Composer Si Window Help * - @	Audio Audio	11. File Options	Splay_error_numeer " ti-cgt-msp43@_18.1.1 ml"use_hw_mpy=16 - lizing all unused por lizing all unused por lizing all unused por lizing all unused por	asm_listing -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa ts to eliminate wa ts to eliminate wa	Quick Access	
 Edit View Navigate Project Run Scripts Console 22 Console 23 Console 24 CDTBuild Console [lbb] D11 -heap size 300stack size -lag wrap=offdisplay_err -it.c; rit.csv8/cools/compiler -diag wrap=offdisplay_err -heap size (ULP 1.1) De consumption on unused pins. remark #10372-D: (ULP 4.1) De consumption on unused pins. remark #10372-D: (ULP 4.1) De consumption on unused pins. remark #10372-D: (ULP 4.1) De consumption on unused pins. 	D1.map - Code Composer St Window Help	Audio Audio	11. File Options	Spidy_error_numeer " "t-cgt-msp43@_18.1.1. ml"use_hw_mpy=16 - lizing all unused por lizing all unused por lizing all unused por lizing all unused por lizing all unused por	asm_listing -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa ts to eliminate wa ts to eliminate wa ts to eliminate wa	Quick Access	
 Edit View Navigate Project Run Scripts Console Console Conso	D1.map - Code Composer St Window Help * * * * * * * * * * * * * * * * * * *	tudio tudi	11. File Options	Spiay_error_numeer " "t-cgt-msp43@_18.1.1. ml"use_hw_mpy=16 - lizing all unused por lizing all unused por	asm_listing -2 -m LTS/include"rer -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
<pre>le Edit View Navigate Project Run Scripts le Edit View Navigate Project Run Scripts le Console Le Console</pre>	_Dl.map - Code Composer Si Window Help ** * @ * ! @ * ! @ * * * ******************************	A GLE AND	11. File Options	Splay_error_numper '' ti-cgt-msp430_18.1.1. ml"use_hw_mpy=16 - lizing all unused por lizing all unused por	asm_listing -2 -m LTS/include"rer -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
e Edit View Navigate Project Run Scripts • Console 20 <td>_DI.map - Code Composer Si Window Help ** * @ * !@ > * < < printsupport = Tui printsupport = Tui printsupport = Tui sonsupport = -vansec cinit_hold_wdt=</td> <td>itudio itudio</td> <td>11. File Options</td> <td>Syliay_error_numeer '' 'i-cgt-msp43a_18.1.1. ml"use_hw_mpy=16 - lizing all unused por lizing all unused por</td> <td>asm_listing -2 -m LTS/include"rer -rom_model -o "Lab ts to eliminate wa ts to eliminate wa</td> <td>Quick Access</td> <td></td>	_DI.map - Code Composer Si Window Help ** * @ * !@ > * < < printsupport = Tui printsupport = Tui printsupport = Tui sonsupport = -vansec cinit_hold_wdt=	itudio itudio	11. File Options	Syliay_error_numeer '' 'i-cgt-msp43a_18.1.1. ml"use_hw_mpy=16 - lizing all unused por lizing all unused por	asm_listing -2 -m LTS/include"rer -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
<pre>le Edit View Navigate Project Run Scripts Console 22 Console 23 Console 23 Console 23 Console 24 Console</pre>	_DI.map - Code Composer Si Window Help *** @ * @ * @ * < *** @ * @ * @ * *******************	itudio itudio	11. File Options	Splay_error_numper ''	asm_listing -2 -m LTS/include"rem -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
le Edit View Navigate Project Run Scripts le Edit View Navigate Project Run Scripts le Console 20 <	_DI.map - Code Composer Si Window Help ====================================	kudio kudio	11. File Options	Splay_error_numper ''. ''. ''. ''. ''. ''. ''. ''. ''. ''	asm_listing -2 -m LTS/include"rem -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
le Edit View Navigate Project Run Scripts Console 23 Console 23 Console 23 Console 23 Console 24	DI.map - Code Composer Si Window Help ************************************	itudio itudio	11. File Options () () () () () () () () () () () () () (Splay_error_numeer	asm_listing -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
Le Edit View Navigate Project Run Scripts Console 22 OTB Built Console [Lab1.D1]ULTITE=CTAPASTONDIA DTB Built Console [Lab1.D1]	DI.map - Code Composer Si Window Help ************************************	itudio itudio	11. File Options	Splay_error_numeer	asm_listing -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
 Edit View Navigate Project Run Scripts Console 	DI.map - Code Composer Si Window Help ************************************	itudio itudio	11. File Options	Splay_error_numeer	asm_listing -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	
Edit View Navigate Project Run Scripts William Schemer Schemer Console 22 Console 22 C	DI.map - Code Composer Si Window Help ************************************	itudio itudio	11. File Options	Splay_error_numeer	asm_listing -2 -m LTS/include"ren -rom_model -o "Lab ts to eliminate wa ts to eliminate wa	Quick Access	

Figure 12. CDT Build Console

Step 4. Repeat the steps to build the "twofact.c" file. Click the ficon to build and link the project.

Step 5. Open the Debug folder in the Project Explorer to see the generated list files. The list files have the extension .lst. For example, twofact.lst is generated containing a C source code with the corresponding assembly language code for twofact.c source file. The files with extension .obj will be used as inputs to the linker. The debug executable has the extension .out and will be used as an input to the Debugger.

Step 6. Open the map file from the debug folder and examine the different sections and the space they will use in memory. The formatting can take some getting used to but there are several tools available for graphical analysis, including one inside of CCS. From the menu bar select "Memory Allocation" (Figure 13). Examine the Memory Allocation table to see which sections are placed into which type of memory (Figure 14).

	325 - C	CS Edit - Lab_D1	/Debug/Lab_D	1.map - (Code Comp	1051
	Viev	v Navigate P	Project Run	Scripts	Window	H
	0	Resource Explo	orer			þ
	0	Resource Explo	orer Classic			
	ab	Grace Snippets	5			
	8	Getting Started	ł			
) V	CCS App Center	er			
	Q	GUI Composer	тм		>	
	ß	Project Explore	er			
	8	Problems		Alt+Sł	nift+Q, X	
1		Console			nift+Q, C	
	8	Advice				
	*	Debug				
	0	Memory Brows	ser			
	1010 0101	Registers				
	କୁଟ୍ର	Expressions				
	(×)=	Variables		Alt+Sł	nift+Q, V	
	===··	Disassembly				
	•	Breakpoints		Alt+Sł	nift+Q, B	
		Modules				
		Terminal				
	1	Scripting Cons	ole			
	2	Target Configu	urations			
	<u>=</u>	Outline		Alt+Sh	ift+Q, O	
		Stack Usage				
		Memory Alloca				
		Optimizer Assi	stant			
		Other		Alt+Sh	ift+Q, Q	

Figure 13. View Menu

Hemory Allocation 8	3					× E) 🕞 📑 🛃 🔻 🗖
Project 'Lab_D1': Link succ	essful						
PERIPHERALS_8BIT							240
PERIPHERALS_16BIT							256
INFOB							128
INFOA							128
V RAM	1,250 (15%)						8,192
> .data			404				
> .sysmem		300					
> .cio		288					
> .bss	178						
> .stack	80						
V FLASH	556 (1%)						52,926
> .binit							
> .text:_isr	36						
> .const					398		
> .cinit		122					
✓ FLASH2		17,532 (26%)					65,536
> .text							17,532

Figure 14. Memory Allocation Table

Step 7. You can also examine the estimated stack usage by selecting the "Stack Usage" option from the View menu. Change the optimization settings and example how the Memory Allocation and Stack Usage change from the various options. Reset the optimization level back to the off setting.

2 Debugging

This section continues the development cycle started in the previous section and explores the basic features of the Debugger.

2.1 Starting the Debugger

The correct debugger options were setup during the project creation, TI MSP430 USB1, and will work from both the Experimenter board and the Launchpad. Make sure the correct device is connected to your workstation.

Step 1. Click the icon from the menu bar to start debugging.

Step 2. In the UPL Adviser Dialog (Figure 15) click the <Proceed> button.



Figure 15. ULP Advisor

Step 3. After downloading the code, the application will be halted at the start of main as shown in Figure 16.



Figure 16. Debugger Entry Location

2.2 Inspecting Source Statements

The following debugging options are available for inspecting the code while it runs on the target hardware.

- Resume (F8). Run the program.
- Terminate <a>

 (CTRL + F8). Stop execution of the program.
- Suspend III (Alt + F8). Pause execution of the program.
- Step Into 🎿 (F5). Step into a function.
- Step Over 🍄 (F6). Step over the next line.
- Step Out 📽 (F7). Step out of a function.
- Restart 🕗. Start execution of the program from the beginning.

Step 1. Use the Step Over command until you have passed the ifact function as shown in Figure 16.

```
19 int main(void)
20 {
      // Stop watchdog timer to prevent time out reset
21
      WDTCTL = WDTPW + WDTHOLD;
22
23
      int i;
24
      int a;
25
      long int b;
26
      printf("( i)! : int-dec, int-hex : long int-dec, long int-hex;\n");
27
      for (i = 1; i < MAXF; i++) {</pre>
28
          a = ifact(i); // Call subroutines to do the calculation
29
          b = lifact(i);
30
31
          printf("(%2d)! : %8d, %8x : %14ld, %14lx;\n", i, a, a, b, b);
32
      }
33
      return 0;
34 }
```

Figure 16. Passing the ifact function

2.3 Inspecting Variables

CCS allows you to watch variables or expressions in the source code, so that you can keep track of their values as you execute your application. You can look at a variable in many ways. For example, you can view a variable by pointing at it in the source window with the mouse pointer, or by opening one of the Variables, or Watch windows.

Note: When optimization level off is used, all non-static variables will live during their entire scope and thus, the variables are fully debuggable. When higher levels of optimizations are used, variables might not be fully debuggable. A good rule of thumb is to increase optimization until the program meets design constraints. For this lab turning off optimization will meet these constraints. By default, CCStudio shows you the Variables window which gives you a list of the current variables in the scope of the executing code (e.g., Figure 17).

(x)= Variables 🔀 😚	Expressions 1010 Regis	sters		£_ ⇒t: 🕒 📋
Name		Туре	Value	Location
(×)= a		int	1	Register R9
(×)= b		long	1	R13:16,R12:16
(×)= i		int	1	Register R10

Figure 17. Local Variables Window

2.3.1 Setting a Watchpoint

Next you will use the Watch window to inspect variables.

Step 1. Double click one of the variables to highlight it in the source code. Next right-click it and select "Add Watch Expression..." (Figure 18).

b	Breakpoint (Code Composer Studio)	, aluc
р	Open Declaration	F3 , i,
} ret	Cut	Ctrl+X
	Сору	Ctrl+C
	Paste	Ctrl+V
~	Use Spaces for Tab	
	Declarations	>
	References	>
	Search Text	>
=⇒]	Run to Line	Ctrl+R
nsole 🦜	Move to Line	
CIO x+y ₽	Add Watch Expression	int
	Preferences	1110-

Figure 18. Text Options Menu

Step 2. In the Add Watch Expression Dialog box (Figure 19) click the <OK> button.

		A
5	X+9 Add Watch Expression X	
10 0	Expression to watch:	
	a	
	OK Cancel	V V

Figure 19. Add Watch Expression Dialog

The variable should now appear in the Watch Expressions Window as shown in Figure 20.

(x)= Variables 🚱 Expressions 🔀	1010 Registers	1	‱ ≈4 🕞 🕂 💥 🔆 🔂 🖬 🖆 🖉 ∨ 🖓			
Expression	Туре	Value	Address			
(×)= a	int	1	Register R9			
🐈 Add new expression						

Figure 20. Watch Expressions Window

The Continuous Refresh button can also be used to update the value while the program is running.

2.3.2 Setting and Monitoring Breakpoints

Breakpoints can be added to the code by double-clicking the blue shaded area next to the line number where you want the breakpoint to stop (e.g., Figure 21 Line 28).



Figure 21. Breakpoint in code (Line 28)

2.3.3 Executing up to a Breakpoint

Pressing the Resume button or F8 will run the program until the breakpoint is reached.

2.4 Debugging in Disassembly Mode

Debugging in the disassembly can be done by selecting "View->Disassembly" (Figure 22) from the menu bar.

*	120 - 00	S Debug - Lab_D I/	rabi_birc - O	ode Compos	er stu	
·	View	Project Tools	Run Script	s Window	Help	
511) 🧭	Resource Explorer Resource Explorer				
5	b V	Getting Started CCS App Center				
100	Θ	GUI Composer™			>	
LOB		Project Explorer Problems Console Advice		Alt+Shift+Q, Alt+Shift+Q,		D/A
	1	Debug Memory Browser Registers Expressions				
	(×)=	Variables		Alt+Shift+Q,	v	
		Disassembly				
	- 🛋	Breakpoints Modules		Alt+Shift+Q,	В	
	'C					[

Figure 22. View Menu

In this view as shown in Figure 23, you can see how the compiler has translated your C code to the machine language of the processor.

	(E) (E) : 4	tom tom ton ton		
Σ,		▼ : <□ : ⊑ : I▶ II I		.e 🖩 📴 % 🖉 • 🕅 🕅 🛸 • 🕹 🗉 • 🔯 • 🗷 🍕 Ø 🔗 •
•••	w Disassem	bly S2		
5		•		
*	50	if (neg) *bufp		P10
	006680: 006682:	930A 2404	TST.W	R10
8	006684:	8318	JEQ	(\$C\$L3) R8
	006686:	40F8 002D 0000	DEC.W MOV.B	
	52			#0x002d,0x0000(R8) 1 = (tempc + BUFLEN) - bufptr);
	52	\$C\$L3:	iper, uva	I = (Cempc + BOFLEN) - Bulpch);
8	00668c:	410A	MOV.W	SP.R10
	00668e:	880A	SUB.W	R8,R10
	006690:	503A 0014	ADD.W	#0x0014,R10
	006694:		MOV.W	R9.R12
	006696:		MOV.W	R8,R13
	006698:	4A0E	MOV.W	R10,R14
	00669a:	12B0 75D2	CALL	#memcpy
	53	return uval - 1;	/* DO	N'T COUNT NULL TERMINATION */
	00669e:	4A0C	MOV.W	R10,R12
	0066a0:	831C	DEC.W	R12
	0066a2:	5031 0014	ADD.W	#0x0014,SP
	0066a6:	4030 7608	BR	<pre>#mspabi_func_epilog_5</pre>
	20	{		
		main():		
	0066aa:	8031 0018	SUB.W	#0x0018,SP
	22	WDTCTL = WDTPW +		
	0066ae:	40B2 5A80 0120	MOV.W	#0x5a80,&Watchdog_Timer_WDTCTL
	26			<pre>c, int-hex : long int-dec, long int-hex;\n");</pre>
	0066b4:	40B1 776E 0000	MOV.W	#0x776e,0x0000(SP)
	0066ba:	12B0 71C4	CALL	#printf
	27	<pre>for (i = 1; i <</pre>		
	0066be:	4391 0014	MOV.W	#1,0x0014(SP)
	0066c2:	90B1 0010 0014	CMP.W	#0x0010,0x0014(SP)
	0066c8:	342E	JGE	(\$C\$L2)
	28		; // Ca	ll subroutines to do the calculation
	Ø066ca:	\$C\$L1: 411C 0014	MOV.W	0:0014(SD) D12
	0066ce:	1280 6580	CALL	0x0014(SP),R12 #ifact
	0066d2:	4081 0016	MOV.W	#1TACT R12,0x0016(SP)
	29	4C81 0016 b = lifact(i		MIZ JOYOOTO (JF)
	29 0066d6:	411C 0014	MOV.W	0x0014(SP),R12
	0066da:		CALL	#lifact
	0066de:		MOV.W	#11/acc R12,0x0010(SP)
	0066e2:	4081 0012	MOV.W	R13,0x0012(SP)
	31	printf("(%2d		d, %8x : %14ld, %14lx;\n", i, a, a, b, b);
	0066e6:	40B1 77AF 0000	MOV.W	#0x77ae.0x0000(SP)

Figure 23. Disassembly View

2.5 Monitoring Registers

The Register window lets you monitor and modify the contents of the processor registers. Notice registers PC (Program Counter), SP (Stack Pointer), SR (Status Register), and R4-R15 (general-purpose registers) in Figure 24.

x)= Variables 60 Expressions 100 Registers 🔀		£
Name	Value	Description
✓ ₩ Core Registers		Core Registers
1919 PC	0x013A14	Core
10101 SP	0x0030E2	Core
> 1919 SR	0x0004	Core
1919 R3	0x000000	Core
10101 R4	0x08BF7B	Core
1919 R5	0x000068	Core
1919 R6	0x000000	Core
1010 R7	0x01FFEF	Core
1111 R8	0x0FFFFD	Core
1919 R9	0x000006	Core
1919 R10	0x000003	Core
1919 R11	0x000003	Core
100 R12	0x000006	Core
1010 R13	0x000003	Core
1010 R14	0x000025	Core
100 R15	0x000004	Core

Figure 24. Registers Window

Step 1. Step Over to execute the next instructions and watch how the values change in the Register window.

Step 2. Close the Register window. To see the cycle count, click on Run->Clock->Enable (Figure 25) from the debug perspective.

Ru	n Scripts Window H	lelp			
: 🖬	Connect Target	Ctrl+Alt+C	6	N - 🖉 🖏	1 4
	Disconnect Target	Ctrl+Alt+D	E		
36	Restore Debug State	Alt+E			*
¦ 🖉	Load	>	Ŀ		
) 🗈	Resume	F8			
	Suspend	Alt+F8	Ŀ		
	Terminate	Ctrl+F2	Ŀ		
6-9	Disconnect		Ŀ		
Q	Go Main	Alt+M	Ŀ		
۵	Reset	>	Ŀ		
2	Restart		Ŀ		
3.	Step Into	F5	Ŀ		
9	Step Over	F6	Ŀ		
Э.	Assembly Step Into	Ctrl+Shift+F5	Ŀ		
- 🔿	Assembly Step Over	Ctrl+Shift+F6	Ŀ		
- P2	Step Return	F7	Ŀ		
=>]	Run to Line	Ctrl+R	Ŀ		
	Free Run	Ctrl+F8	Ŀ		
	Step Into Selection		L		
1	Clock	>	~	Enable	
r	Advanced	>		Disable	
1 🐀	Debug	F11	~	Show	
- k	Debug History	>		Hide	
e 2	Debug As	,		Reset	
4	Debug As Debug Configurations	>		Setup	
-	Debug Configurations.		1		

This icon $(\underbrace{ \bigcirc :2}_{ })$ will show in either the bottom of CCS or in the status bar. More information about the profile clock can be found at

http://processors.wiki.ti.com/index.php/Profile clock in CCS

2.6 Monitoring Memory

The Memory window (Figure 26) lets you monitor selected areas of memory. You can select RAM, flash, or SFR portion of the memory.

Disassem	bly 🚺 Memory Browser 🔀 📟	
	🤹 - 🖗 - 🚸 🍫 📑 🖻	∇
0	5 7	
))x0 - 0 <men< td=""><td>nory Rendering 1> 🛛</td><td></td></men<>	nory Rendering 1> 🛛	
16-Bit Hex -	TI Stule	
0x000000	Special_Function_IE1	^
0x000000	0000	
0x000002	Special_Function_IFG1 2A02.002A 0A0A	
0x000002		
0x000008 0x000008	Port_9_10_P9IN, Port_9_10_PBIN 0000	
0x000008 0x00000A	Port 9 10 P9OUT, Port 9 10 PBOUT	
0x00000A	FBFB	
0x000000C	Port 9 10 P9DIR, Port 9 10 PBDIR	
0x00000C	0000	
0x000000E	Port 9 10 P9SEL, Port 9 10 PBSEL	
0x00000E	0000 0000 0000 0000 0000	
0x0000018	Port 3 4 P3IN	
0x000018	CA20	
0x00001A	Port 3 4 P3DIR	
0x00001A	0000	
0x00001C	Port 3 4 P4IN	
0x00001C	1800	
0x00001E	Port 3 4 P4DIR	
0x00001E	0000	
0x000020	Port 1 2 P1IN	
0x000020	3E03	
0x000022	Port_1_2_P1DIR	
0x000022	0000	
0x000024	Port 1 2 PIIES	
0x000024	005A	
0x000026	Port_1_2_P1SEL	
0x000026	FF00	
0x000028	Port_1_2_P2IN	
0x000028	7820	
0x00002A	Port_1_2_P2DIR	
0x00002A	0000	

Figure 26. Memory View

Step 1. To open the Memory Browser, select "View->Memory Browser" (Figure 27) from the menu bar.

	ies des bebug ess_billes ies ies reade composer or	
	: View Project Tools Run Scripts Window Hel	
1.2	Image: Weight of the second	SA
	GUI Composer™ > Project Explorer	
	Memory Browser Registers	

Figure 27. View Menu

Step 2. If all of the memory units have not been initialized yet, continue to step over and you will notice how the memory contents will be updated.

Step 3. You can change the memory contents by editing the values in the Memory window. Just place the insertion point at the memory content that you want to edit and type the desired value.

Step 4. Close the memory window.

2.7 Viewing Terminal I/O

Sometimes you might need to debug instructions in your application that make use of stdin and stdout without the possibility of having hardware support. CCS lets you simulate stdin and stdout by using the Console (e.g., Figure 28).

Note: While the correct settings were set above for printf support in this program, some programs may need more heap memory to correctly print to the console. More information about printf can be found at http://processors.wiki.ti.com/index.php/Tips for using printf

- I- D1-C	10					
.ab_D1:C	.10					
(i)!	:	int-dec,	int-hex	:	long int-dec,	long int-hex;
(1)!	:	1,	1	:	1,	1;
(2)!	:	2,	2	:	2,	2;
(3)!	:	6,	6	:	6,	6;
(4)!	:	24,	18	:	24,	18;
(5)!	:	120,	78	:	120,	78;
(6)!	:	720,	2d0	:	720,	2d0;
(7)!	:	5040,	13b0	:	5040,	13b0;
(8)!	:	-25216,	9d80	:	40320,	9d80;
(9)!	:	-30336,	8980	:	362880,	58980;
(10)!	:	24320,	5f00	:	3628800,	375f00;

Figure 28. Console Output of Execution Program

Note: The contents of the window depend on how far you have executed the application.

3 Software Documentation

Maintaining good software documentation is of key importance, especially when your code may be used by others. A special focus should be placed on your header and comments, code formatting, and software diagrams or flowcharts.

3.1 Code Formatting and Organization

Part of good programming is ensuring that the code is neatly formatted and organized. This not only helps others who may need to access your code, but it also helps in debugging and maintaining your own code. Here are some general guidelines to assist in organizing your code in the Code Composer Studio IDE:

- Ensure all your function prototypes are declared after your #include statements at the top of the program.
- Declare all your global variables directly after your function prototypes in one area
- Consistently organize your functions. A good way to organize functions is to have your main function, followed by the other functions in order of call, followed by your interrupt functions. You can choose a different way, but make sure to organize the types of functions and maintain consistency.

- Keep track of your indention. In the CCStudio IDE, a tab is four spaces. Each function, loop, or other "nest" should be indented appropriately and consistently.
- It is especially important to keep track of where your code is located on your workstation. CCStudio uses workspaces to help you keep track of your code. It is recommended that you create a directory where you will keep all your projects. Create subdirectories for each lab assignment. Your code is your creative expression and thus take care of it. Your engineering reputation will depend on the quality of your code.

3.2 Code Headers and Comments

By now, you have become familiarized with commenting in your programs. Comments help you and others keep up with the flow of the code, and it is important to maintain good comments. In this Laboratory, you will be programming in C and assembly language. In assembly language, you generally should comment every line of code to explain its purpose. The reason for this is that the code is much less self-explanatory than common coding languages. In C, there are generally a few guidelines to remember when commenting:

- You should always include a header at the top of your code that gives basic information about your program (what it does and how), author, and when the code was written.
- Each variable declaration should be commented.
- Each function declaration should be properly noted.
- Any segment of code, whether it is to initialize hardware, perform a calculation, or do another task, should have concise comments that explain it.

3.3 Software Flowcharts

A flowchart is a helpful way for you to decide on an approach to your program before you begin. It is also an extremely effective way of concisely relaying how your code works to others. A flowchart does not contain information about every line of code, but it is a slightly higher-level picture that shows logically how problems are addressed. Hardware initializations and variable declarations should be documented. Also, any logical steps, function calls, or loops should be noted as well as their respective conditions.

A flowchart for calculating the factorial can be seen below in Figure 29. Note that it does not include every line of code, but it does capture the main steps in the program.

