

Let's Get Started!

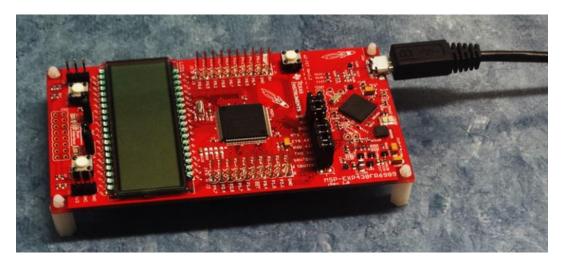
This handout will walk you through the steps to create your first program on the Texas Instruments MSP430FR6989 Microcontroller Launchpad.

Note, we go through a lot of details in these lab manuals, and at times, some students have thought we included too many steps. However, it is our intent to err on the side of caution and provide as much support for our students as possible. Thanks for understanding.

1. Open the Launchpad development kit box and the plastic bag that contains the Launchpad board.



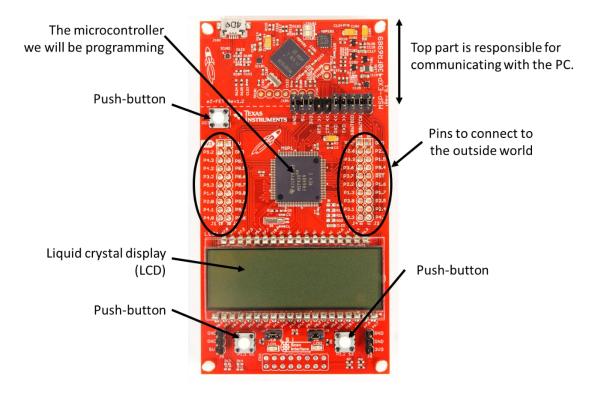
2. Connect the Launchpad development board to the PC with the enclosed USB cable.





3. The board is divided into two parts. The top part (the side with the USB connector) is responsible for communicating with the PC make getting your program onto our MSP430FR6989 microcontroller.

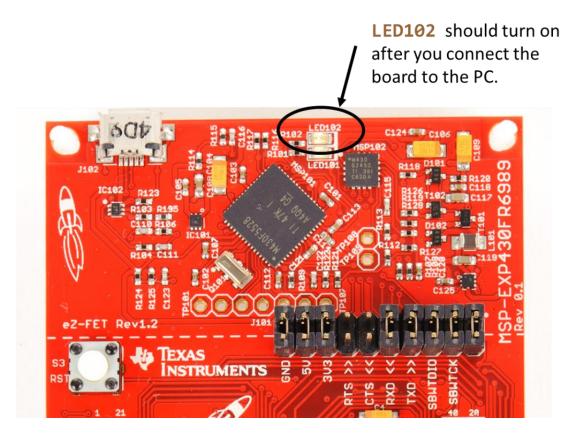
The bottom part contains the MSP430FR6989 that we will be using in this class. In addition, the bottom part has the LCD screen, push-buttons, and a number of metal pins that can be used to connect the microcontroller to the outside world. Each of these will be discussed in more detail later in the course.





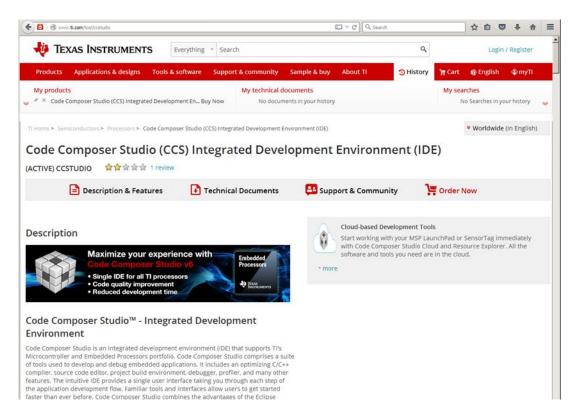
4. The power LED (LED102) should be on after you connect the board to the PC. The bottom twothirds are for the microcontroller, two push-buttons, a couple LEDs, and holes for connecting your board to the rest of the world. Note, there might be slight differences in the printing on the board. As the board goes through revisions, the printing is occasionally changed.

Note, the font type for **LED102**. Periodically, you will see words and phrases using a similar format throughout our documentation. This will be your sign that this is either an important reference from the Texas Instruments documentation or hardware / software tools.



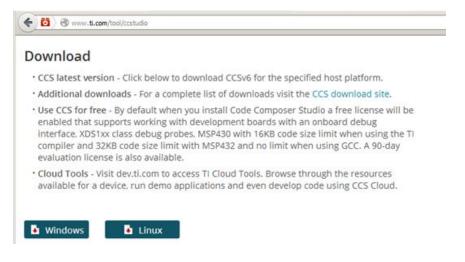


5. If you have not done so already, follow this <u>link</u> to download the **Code Composer Studio** (**CCS**) program.



http://www.ti.com/tool/ccstudio

6. Scroll down the page a little and you will see a link to download **CCS**. We will be using the Windows version of the program. Go ahead and click the **Windows** button.





7. At this time, you will need to either create your Texas Instruments account or, if you have an account, enter your **email address** and **myTI** password.

Texas Instruments	Everything * Search		2		
lucts Applications & designs	Tools & software Support & commun	nity Sample & buy About TI	ोन Cart	🔮 English	¢ my
TI Account					
Existing myTl user?	New user? Register for free:				
Your email address	Country or region	💸 my T			
Your myTI password	Select one	 Buy ICs, tools & software 			
Remember me 🚯	Zip or postal code	Request samples			
Login	Company	 Get support on E2E 			
Forgot your password?	First name	 Simulate designs in WEBENCH[®] 			
	Last name	 Set alerts for products & software 			
	Your email address	 Personalize your web experience 			
	Confirm email address				
	Create a password				
	Confirm password				

8. After logging in, you will need to complete the U.S. Government export approval form. Enter your information, and click **Submit** at the bottom of the form.

🐺 Texas Instruments	Everything * Search			م	Hello, Mar	k Logout
Products Applications & designs Tools a My products * × Code Composer Studio (CCS) Integrated Develo Filme		nical documents o documents in your histor		3 History	'ႃၛ Cart	The second secon
l Request	U.S. Governmen	t export approv	val:			
 If you are approved, a DOWNLOAD BUTTON will appear. If you are not immediately approved, a message will appear after this form: Software may be delayed 1-2+ business days. This file requires U.S. Government export approval before download. To AVOID delays, please provide complete information and do NOT use abbreviations We apologize for any inconvenience. 	First name: Last name: Your email address: Your full company/university	United States	du reviate. Full legal			



9. After a few moments, you will be taken to the download page. Click **Download** to start the process.

Verything	* Search		Q		Hello, Mark	Logout	
Products Applications & designs Tools & software	Support & community	Sample & buy Abo	out TI 3 History	ोस Cart	🧐 English	∲myT I	
My products	My technical docu Now No document	iments ts in your history		My sea	arches No Searches in yo	our history	*
TI Home							
TI Request							
You have been approved to receive this f Click "Download" to proceed.	file.						
In a few moments, you will also receive an email with the link to t	this file.						
Download Having trouble downloading? Try www.ti.com/software-help							
Thank you, Texas Instruments							

10. When complete, you will have a 7.3MB file entitled **ccs_setup_win32.exe**. Go ahead and open the file.





11. Next, you will need to accept the terms of the license agreement and click **Next**.

😚 Code Composer Studio v6 Setup
License Agreement Please read the following license agreement carefully.
Code Composer Studio 6.1 Software License Agreement
IMPORTANT – PLEASE READ THE FOLLOWING LICENSE AGREEMENT CAREFULLY. THIS IS A LEGALLY BINDING AGREEMENT. AFTER YOU READ THIS LICENSE AGREEMENT, YOU WILL BE ASKED WHETHER YOU ACCEPT AND AGREE TO THE TERMS OF THIS LICENSE AGREEMENT. DO NOT CLICK "I ACCEPT" UNLESS: (1) YOU ARE AUTHORIZED TO ACCEPT AND AGREE TO THE TERMS OF THIS LICENSE AGREEMENT ON BEHALF OF YOURSELF AND YOUR COMPANY; AND (2) YOU INTEND TO ENTER INTO AND TO BE BOUND BY THE TERMS OF THIS LEGALLY BINDING AGREEMENT ON BEHALF OF YOURSELF AND YOUR COMPANY.
Important – Read carefully: This Code Composer Studio 6.1 Software License Agreement ("Agreement") is a legal agreement between you (either an individual or entity) and Texas Instruments Incorporated ("TI"). The "Licensed Materials" subject to this Agreement include the software programs (in whole or in part) that accompany this Agreement and any "on-line" or electronic documentation (in whole or in part) associated with these software programs. By installing, copying or otherwise using the Licensed Materials you agree to abide by the provisions set forth herein. This Agreement is displayed for you to read prior to using the Licensed Materials. If you choose not to accept or agree with these provisions, do not download or install the Licensed Materials.
Note Regarding Access to Open Source Software: The Licensed Materials are bundled with open source software. By accepting this Agreement, you will gain access to software licensed under open source licenses, in which case
• I accept the terms of the license agreement.
C I do not accept the terms of the license agreement. Print Texas Instruments
< Back Next > Finish Cancel

12. You will be asked to specify the location for installation. **Browse** to your preferred location or accept the default and click **Next**.

💱 Code Composer Studio v6 Setup				×
Choose Installation Location Where should Code Composer Studio v6 be ins	talled?			
To change the main installation folder clic	k the Browse	button.		
CCS Install Folder				
c:\ti				Browse
-				
Texas Instruments				
ar service at rese of the 136	< Back	Next >	Finish	Cancel



13. You will then be asked to select which microcontrollers and microprocessors you would like to use with CCS. For now, just select the MSP Ultra Lower Power MCUs as shown below and click Next.

💱 Code Composer Studio v6 Setup	×
Processor Support Select Product Families to be installed.	
Image: State of the state	Description Processor Architectures included: MSP430, MSP432
C Select All	Install Size: 977.84 MB. Download Size: 325.95 MB.
Texas Instruments	<back next=""> Finish Cancel</back>

14. The next window asks you to **Select Debug Probes**. Select the options shown below and click **Next**.

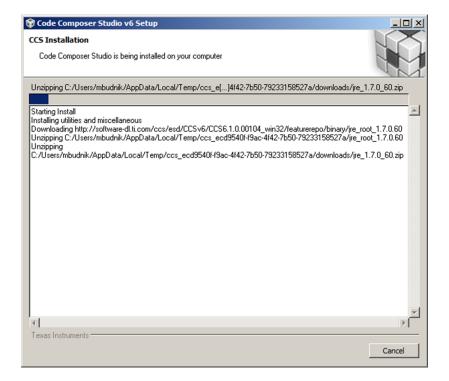
🞲 Code Composer Studio v6 Setup	×
Select Debug Probes Select the debug probes you want installed and deselect	the debug probes you want to leave out.
	Description
☑ TI XDS Debug Probe Support	Description
Blackhawk Debug Probes	
Spectrum Digital Debug Probes and Boards	
MSP430 USB FET	
MSP432 USB FET	
_	
C Select All	Install Size: 1236.22 MB.
	Download Size: 412.07 MB.
Texas Instruments	
< B:	ack Next > Finish Cancel
< b.	



15. Ok, we are almost done. The last window asks if you want to install any Texas Instrument apps for **CCS**. Do not select any of these options and click **Finish** to begin the installation.

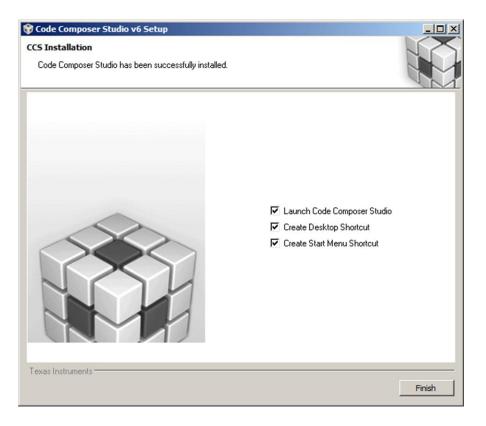
😵 Code Composer Studio v6 Setup	×
App Center Texas Instruments	-
There are additional products and features ("add-ons") available fr add-ons will be installed by the App Center when you run Code Com	
Tools (for App Center background download)	Description
□ □ Software (for App Center background download)	
MSPWare	
GUI Composer	
Select All	Install Size: 1236.22 MB.
	Download Size: 412.07 MB.
Texas Instruments	,
<back next<="" td=""><td>Finish Cancel</td></back>	Finish Cancel

16. Windows will show you the progress of the download. Note, the download and installation may take a couple minutes.

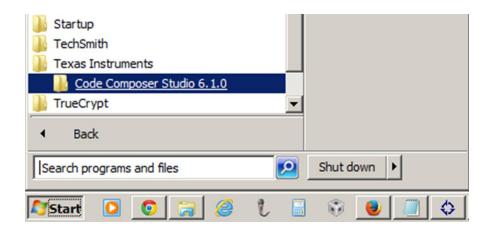




17. When the installation is complete, you will have the option of immediately launching **Code Composer Studio**.



18. In the future, you can open **CCS** from the start menu.





19. Note, there is also an off-line installer option for **Code Composer Studio**.

http://processors.wiki.ti.com/index.php/Download CCS

Download the latest CCS

🔒 Windows 📑 Linux

Version 6.1.0.00104 Release Notes @ Linux Installation Instructions System Hardware Requirements

The above buttons will take you to a form that once complete will download a small installation program. A MyTi login is required to fill out the form, if you don't have one there will be an option to create one. This installation program will allow you to select the features of CCS that you want to install and then will download and install just those items. Full zip images 'are available if you have issues with the web installer.' **Previous versions** are also available. After you have completed installation and run CCS for the first time it generate a free license that works with XDS100, development boards with onboard debug or for MSP430 users a 16KB code size limited license is generated. For more information on CCS licenses please see Licensing - CCS. Please set the Linux Host Support CCSv6 page for device support information.

Code Composer Studio Version 6 Downloads

There are two types of installers:

- <u>Web installers</u> will allow you to perform an install using an installer controlled download process that will only download needed software components. An internet connection is mandatory at install time.
- Off-line installers are a large archive (about 730MB). When you run it you can select the components to be installed. No internet connection is required at install time. The executable can be used for installing multiple local systems.

If you have an issue with the web installer not being able to connect to the internet then please try the off-line installer. If you need to update a computer that does not have internet access then download the offline installer and then transfer it to the computer without access and use the offline installer to update your installation.

Release	Build #	Date	Download	Notes
6.1.0	6.1.0.00104	Feb 25, 2015	Web Installers: Windows ଜ Linux ଜ	New in this release: Enhancements to IDE: Integration with Eclipse v4.4.1 and CDT 8.3. Added support for SimpleLink™ CC26xx and CC13xx MCU platform of devices.
			Off-line Installers: Windows & MD5 & Linux & MD5 &	Added support for on-board USB XDS110 debug (Hercules RM46x Launchpad). Support for GCC for MSP430. MSP EnergyTrace Improved EnergyTrace tool for profiling application's energy consumption, battery lifetime, monitoring internal device states and determining execution hotspots (statistical function profile) Bug fixes.
				Installation: see instructions in README Linux @ file. See also additional Linux Installation Instructions @ The manifest @ lists the software components included in this product. If you wish to update a previous install then please install the update from within CCS instead of downloading this installation and pointing it to your existing install.

20. When **CCS** opens, you will first be greeted by a splash screen.





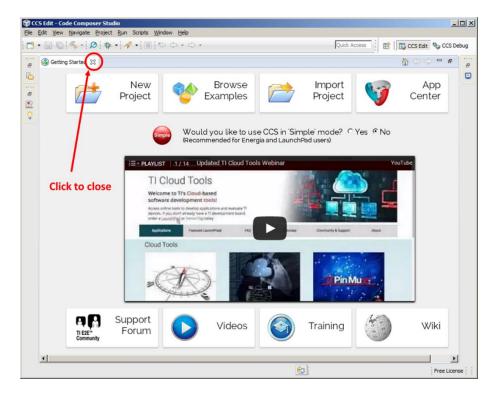
21. You will then be asked to select a **Workspace**. Click **Browse** to specify a folder where you would like to store your **CCS** projects.

Note, make sure you do NOT check the **Use this as the default**... option.

When you are ready, click **OK** to continue.

💱 Workspace Launcher	×
Select a workspace	
Code Composer Studio stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: C:\Users\mbudnik\workspace_v6_1	Browse
□ Use this as the default and do not ask again	
	OK Cancel
🔪 Do NOT check	

22. When opened, the **CCS** program may have a **Getting Started** window open like this. If it does, go ahead and close the tab.





23. When the **Getting Started** tab is closed, **CCS** will look like this. This is the default configuration that you will be using for almost all of your code creation.

CCS Edit - Code Composer Studio	n Scripts Window Help			
			Quick Access	CCS Edit 🏷 CCS Debug
Project Explorer 🛛 🗖 🗖			,	
E				
	E Console	🚽 🗏 - 📑 - 🗆 🗖	🖹 Problems 🔀 💡 Advice	~ - 8
	No consoles to display at this time.		0 items Description A	Resource Path
0 items selected		6		Free License



24. Under File, select New, then select CCS Project.

CCS Edit - Code Composer	Studio		
ile Edit View Navigate Pro		Window Help	
New	Alt+Shift+N	CCS Project	Access
Open File		Project	
Close Close All	Ctrl+W Ctrl+Shift+W	C Source File	
		— h [°] Header File	
Save	Ctrl+5	€ Class	
Save As Save All	Ctrl+Shift+S	File from Template	
Revert		Target Configuration File	
Move		DSP/BIOS v5.x Configuration File	
Rename	F2	RTSC Configuration File	
Refresh Convert Line Delimiters To	F5	Ctrl+N	
Print	Ctrl+P		
Switch Workspace Restart		•	
Import		-	
Export			
Properties	Alt+Enter		
1 main.c [lab_12a_heart_solu	ution]		
Exit			



25. This will open the **New CCS Project** window. It is REALLY important to make sure you get the next several steps right, so please be careful.

For Target, select MSP430FRxxx Family from the pull-down menu.

- a) In the box to the right of **Target**, select **MSP430FR6989** from the pull-down menu. (Note, it is very close to the bottom of the list. This is the microcontroller in the **MSP430FRxxx Family** that we will be using in this class.
- b) For Connection, you should accept the default, **TI MSP430 USB1 [Default]**.
- c) For **Project name**, enter **Blink**.
- d) Under **Project name**, check the **Use Default Location** box.
- e) Use the default **Compiler version**.
- Finally, toward the bottom of the window, under Project Templates and examples, select the Empty Project (with main.c).

After double-checking everything, click **Finish** when you are ready to go on.

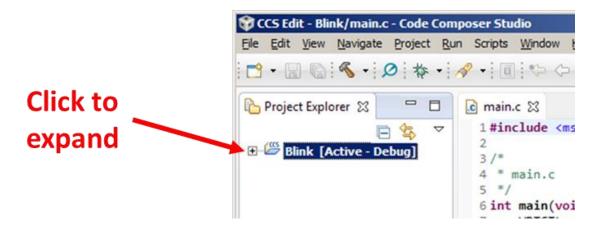
💱 New CCS Project	
CCS Project Create a new CCS Project.	
Iarget: MSP430FRxxx Family Connection: TI MSP430 USB1 [Default] Image: MSP430 Project name: Blink Image: Operation: Compiler version: C:\Users\mbudnik\workspace_v6 Compiler version: TI v4.4.3 Advanced settings Project templates and examples type filter text Empty Projects Empty Project (with main.c) Empty Assembly-only Project Empty RTSC Project Basic Examples Image: Blink The LED Hello World Hello World	MSP430FR6989 Identify Identify Image:
?	Back Mext > Einish Cancel



26. After a few seconds, the new project will be added to the **Project Explorer** window and the "empty" **main.c** file is also shown.

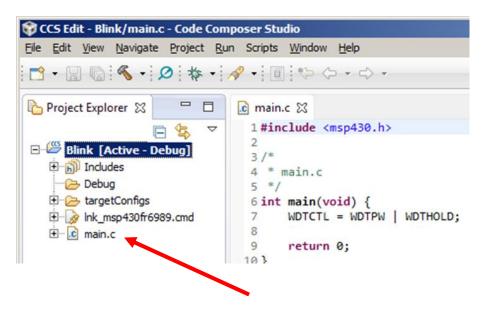
nposer Studio In Scripts Window Help		
	Quick Access 🔡 😰 🕞 CCS	Edit 🏷 CCS Debug
8 9 return 0; 10} 11	g timer	
Gonsole No consoles to display at this time.	Problems 23 🖓 Advice Oitems Description A	Resource Par
	n Scripts Window Help	n Scripts Window Help

27. If you the main.c tab does not open automatically, you can open it anytime you want. First, if the **Blink** folder is collapsed like this, click on the small icon in front of its name.

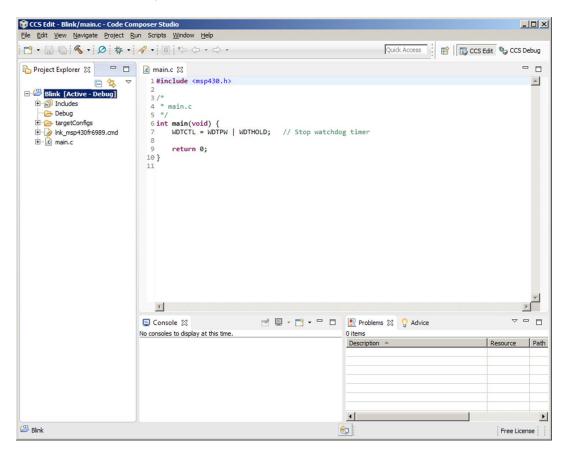




28. This will expand the **Blink** folder to and show its various components. Go ahead and doubleclick on the **main.c** file to open the tab.



29. Make sure your **CCS** window looks like this before proceeding to the next step. If not, please go back and double-check your work.





30. For this handout, the program we will be using is shown below. We will take a look at the various lines of the program in the following steps. These will be covered in a lot more detail later in the course, but for now, our quick explanation should help.

```
// This program will blink a red LED
#include <msp430.h>
                                            // Allows us to use "short-cut" names
                                            // to make our code easier to read
#define
          RED
                       0x0001
                                            // Specifies the red LED light for us
#define
          RED_OFF
                       0x00FE
                                            // Used to turn the red LED off
#define
          ENABLE RED
                       0xFFFE
                                            // Used to enable microcontroller's pins
#define
          DEVELOPMENT
                       0x5A80
                                            // Used to disable some of the security
                                            // features while we are still learning
main()
                                            // All C programs have a main function
{
                                            // Disables some security features
     WDTCTL = DEVELOPMENT;
                                            // Enables the pins to the outside world
     PM5CTL0 = ENABLE RED;
                                            // Make a pin an output
     P1DIR
             = RED;
     long x = 0;
                                            // Will be used to slow down blinking
     while(1)
                                            // Continuously repeat everything below
     {
          for(x=0 ; x < 30000 ; x=x+1);</pre>
                                           // Count from 0 to 30,000 for a delay
          P1OUT = RED;
                                            // Turn red LED light on
          for(x=0 ; x < 30000 ; x=x+1);</pre>
                                           // Count from 0 to 30,000 for a delay
          P1OUT = RED OFF;
                                           // Turn off the red LED light
     }
}
```



31. The first line is a comment that briefly describes what your program will do. To designate a comment, use two backslash characters. **CCS** will recognize the comment and color it green to make it easier to tell apart from the rest of your code.

// This program will blink a red LED

32. The **#include** <msp430.h> command is used to tell **CCS** you want to use, or include, another file when your program is ready. The msp430.h file contains various names of parts of the microcontroller (that is, registers like **P1DIR** and **P10UT**) which can be used to make your program more readable.

Lines like this are not part of the actual program and are often called "preprocessor statements". They are used strictly by the user to specify to **CCS** how the program should be used.

- 33. The next lines are the **#define** statements. These allow you to associate numerical values with names that you can use later inside of your program. Again, this will make your code easier to read.
- 34. The next line signifies the beginning of your actual program. Every C program has a **main()** function. When you run your program, the microcontroller will first look for **main()** and perform everything contained inside of its curly braces **{ }**.

main()

35. In the next line, our program puts the microcontroller into a development or "practice" mode by disabling a security system called the watchdog timer. Essentially, the watchdog timer can be used to restart your program if something goes wrong. This is very important in some more critical programs, but not in simple ones like this.

We will learn more about watchdog timers later, but for now, it simplifies our program to disable it.

Notice also that semicolon (;) at the end of the instruction. You will quickly learn that the C programming language needs to see these semicolons appropriately or your program will not work.

WDTCTL = DEVELOPMENT;



36. The next line of the program is used to enable the microcontroller pins to connect to the outside world. As we go through this course, you may find it amusing to see how many different features of a microcontroller have to be enabled or disabled. However, today's microcontrollers have many, many different features that you could use. By requiring a user to enable the features you want to use, it makes it easier to avoid some mistakes.

For now, just remember that including this line of code in our program allows us to connect the microcontroller to the "outside" world, or in this case, the red LED light.

PM5CTL0 = ENABLE_RED;

37. The next instruction tells the microcontroller to use its pin connected to the red LED light as an output that it can turn on and off.

P1DIR = RED;

38. The next line creates a variable called **x**. This variable will be used to implement a delay so the red LED light will turn on and off slowly enough that you can see it.

Later in the class, we will introduce the term **long** and explain how similar terms can be used to designate how much memory a variable uses in a microcontroller. For now, just recognize that \mathbf{x} can hold values up to about 2,000,000,000.

long x = 0;

39. The next line of the program starts an execution loop.

while(1)

Loops will be discussed in more detail in a later section, but for now know that this loop will keep executing the instructions inside of its curly braces $\{ \}$ indefinitely. Without this loop, the program would only turn the red LED light on and off one time.

40. Next we have our first delay implemented with another loop called a **for** loop.

for(x=0 ; x < 30000 ; x=x+1);</pre>

For loops will also be discussed in more detail in a later section. For our purposes, it creates a delay in our program (set by the 30,000 number). Later in this handout, you can play with increasing/decreasing the length of the delay by changing this number. (But, do not do it yet!)



41. The next instruction turns on the red LED light.

P10UT = RED;

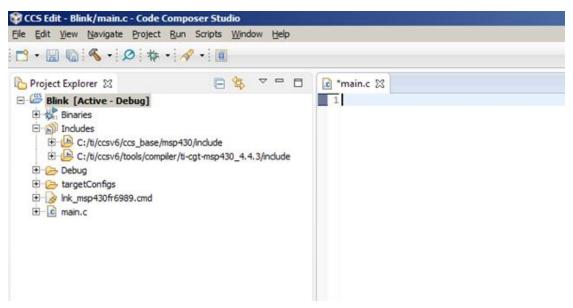
42. We then implement a second delay with another for loop.

for(x=0 ; x < 30000 ; x=x+1);</pre>

43. After the second delay, we turn the red LED light off.

P1OUT = RED_OFF;

- 44. At this point in the program, the while loop returns us back to the previous delay statement to turn the red LED light on and then off again indefinitely.
- 45. Now that we have looked at the program in more detail, let's get it running on your microcontroller. In **CCS**, highlight the program presently displayed in the **main.c** tab. After highlighting it, hit the **[Backspace]** key to delete it. CCS should then look like this:

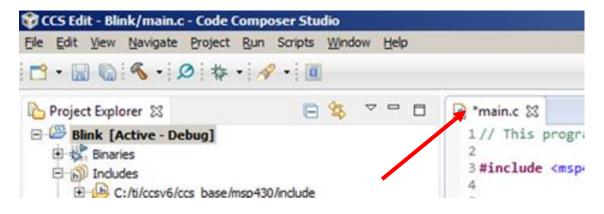




46. Highlight and **Copy** the original, complete program listing from above and **Paste** it into the **main.c** tab.

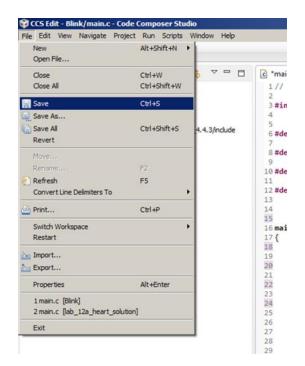
Project Explorer 💥 🗖 🕏 🔻 🗖 🗖		
	€ main.c 🔀	
Blink [Active - Debug]	1// This program will blink a red LED	
E S Includes	3 #include <msp430.h></msp430.h>	// Allows us to use "short-cut" names
E Co Debug	4	<pre>// to make our code easier to read</pre>
E Configs	5 6 #define RED 0x0001	// Constitute the and ISD light for us
🗉 😡 lnk_msp430fr6989.cmd	7	<pre>// Specifies the red LED light for us</pre>
⊞ ic main.c	8 #define RED_OFF 0x00FE	// Used to turn the red LED off
	10 #define ENABLE_RED 0xFFFE	<pre>// Used to enable microcontroller's pins</pre>
	12 #define DEVELOPMENT 0x5A80 13 14 15	<pre>// Used to disable some of the security // features while we are still learning</pre>
	15 16 main() 17 {	// All C programs have a main function
	18 WDTCTL = DEVELOPMENT; 19	<pre>// Disables some security features</pre>
	20 PM5CTL0 = ENABLE_RED; 21	<pre>// Enables the pins to the outside world</pre>
	22 P1DIR = RED; 23	// Make a pin an output
	24 long x = 0; 25 26	// Will be used to slow down blinking
	27 while(1) 28 {	// Continuously repeat everything below
	29 for(x=0 ; x < 10000 ; x=x+ 30	1); // Count from 0 to 30,000 for a delay
	31 P10UT = RED; 32	// Turn red LED light on
	i 33 for(x=0 ; x < 10000 ; x=x+	1); // Count from 0 to 30,000 for a delay
	35 P10UT = RED_OFF; 36 } 37 38 } 39	// Turn off the red LED light
	E Console 🔀	🕹 🔂 😫 🖬 🚮 🚝 🚉 🖃 🛡 😁

47. Look carefully at the **main.c** tab. Do you see the asterisk (*) in front of its name? This asterisk will appear whenever you have unsaved changes in your program.



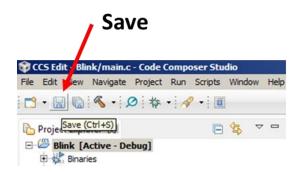


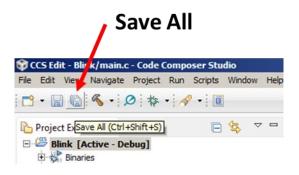
48. To **Save** your changes, you can select **Save** from the **File** menu.



49. Or, you can simply click one of the **Save** shortcut buttons. The first saves only the file (main.c) that is presently opened. The second saves any and all files related to your project (some really big programs may be spread across dozens of files). If you hover your mouse over the **Save** icons, their names will appear.

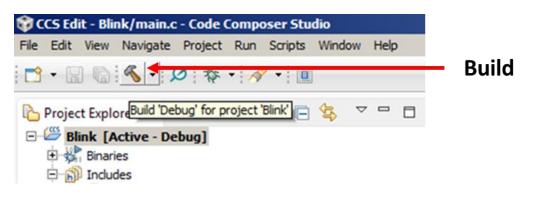
For most of this class, we will only be using a single **main.c** file, so you can use either shortcut.







50. After saving your program, you need to **Build** your project by clicking the hammer icon on the top of your screen. (Get it? You can build things with a hammer?)



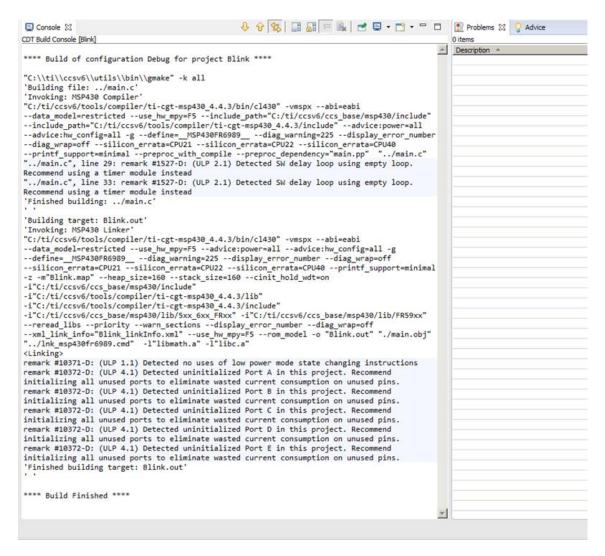
51. After you click **Build**, **CCS** translates your text **main.c** file into the **0**'s and **1**'s that your microcontroller can understand.

💱 Build Project			_ 🗆 🗡
Building project			
Invoking Command: C:\ti\ccsv6	\utils\bin\gmake -k all		
Always run in background			
	Run in <u>B</u> ackground	Cancel	Details >>



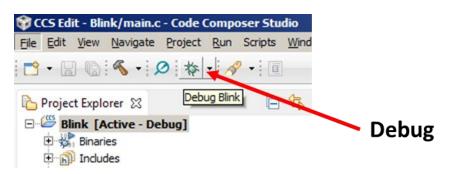
52. The **Build** results will be shown in the **Console** and **Problems** windows.at the bottom of the CCS window.

If there are any errors, you may have accidentally changed some of the instructions in the program. Go back and correct the errors. If CCS reports and warnings, you can disregard them at this time.





53. Once you successfully **Build** your program, you are ready to load it onto the microcontroller. To put your program onto the microcontroller, click the **Debug** button



54. Sometimes when you click the **Debug** button, you'll get an error like this.

If this occurs, make sure your Launchpad is plugged in properly and click **Retry**.

If the error persists try unplugging and then replugging your Launchpad USB cable and then rebuilding your program.

ЭТІ М	SP430 USB1/MSP430		×
8	Error initializing emulator: Interface Communication error		4
		<u>Cancel</u> <u>R</u> etry	



55. You may see the next window pop up. It is notifying you of a new feature in **CCS** that can help you use best practices for ultra-low power applications.

For now, check the **Do not show this message again** box and click **Proceed**.

💱 ULP	Advisor X
1	The Ultra-Low-Power Advisor (ULP Advisor) checks for ultra-low power best practices. You have remarks in you project, which you could use to improve power consumption in your project. Proceed with launch? Cancel to review advice in the Problems View, grouped under the Infos category. Checking is enabled by default. Advice is grouped within the Problems view window under the Infos category.
You have remarks in you project, which you co Proceed with launch? Cancel to review advice Checking is enabled by default. Advice is group To change default ULP Advisor settings, go to For ULP Advisor rule details, visit <u>http://www.</u>	To change default ULP Advisor settings, go to Project > Properties > Build > MSP430 Compiler > ULP Advisor
	For ULP Advisor rule details, visit http://www.ti.com/ulpadvisor.
	For option details, consult the MSP430 Optimizing C/C++ Compiler User's Guide.
Do 1	not show this message again
	[Proceed] Cancel

56. The **CCS Debug** interface (or **Debugger**) will begin loading.

💱 Conf	iguring Debugger (m	nay take a few minute	s on first lau	inch) 🗆 🗙
1	Configuring Debugger	(may take a few minutes	on first launch)
Initializi	ng: MSP430 (Cannot be	canceled)		
	ays run in background			
		Run in <u>B</u> ackground	Cancel	Details >>

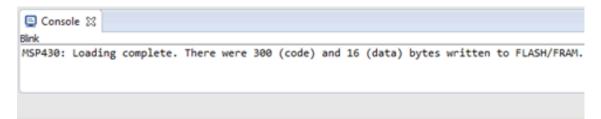


57. When it is loaded, you should see the **CCS Debug** interface. Previously we have been in the **CCS Edit** interface.

As you might expect, you create and edit your programs from the **CCS** Edit interface, and you run your programs from the **CCS** Debug interface.

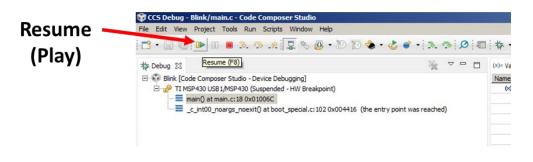
		(x)+ Variables 23 off Expression		Value	(incation) (incation) (incation)	~ -	
E P TI MSP430 US8 1/MSP43	0 (Suspended - HW Breakpo	int)	Name (4)+ x	Type	6682	R15:16,R14:16	
main() at main.c:18 0x01005C into0_noargr_neexit() at boot_special.c:102 0x004416 (the entry point was reached)							
	exit) at boot_speciality to 2	concernence (alle elle y point was reached)					
main.c 23							-
2	blink a red LED						2
3#include <msp430.h> 4 5</msp430.h>		<pre>// Allows us to use "short-cut" nam // to make our code easier to read</pre>	15				
6 #define RED	0x0001	// Specifies the red LED light for	15				
8 #define RED_OFF	0x00FE	// Used to turn the red LED off					
0 #define ENABLE_RED	ØxFFFE	// Used to enable microcontroller's	pins				
1 2 #define DEVELOPMENT 3	ex5A8e	<pre>// Used to disable some of the secu // features while we are still learn</pre>					
4		// Teacures wille we are still lear	ing				
5 6 main()		// All C programs have a main funct	ion				
8 WDTCTL = DEVELO	PMENT;	// Disables some security features					
9 0 PHSCTL0 = ENABLE	_RED;	// Enables the pins to the outside	world				
2 PIDIR = RED;		// Make a pin an output					
1 2 PIDIR = RED; 3 long x = 0; 5 6 7 while(1) 8 { 9 for(x=0; x)		// Will be used to slow down blinking	ng				
while(1)		// Continuously repeat everything b	low				
8 { 9 for(x=0;x	< 10000 ; x=x+1);	// Count from 0 to 30,000 for a del	y.				
1 PIOUT = RED	;	// Turn red LED light on					
2 3 for(x=0; x	< 10000 ; x=x+1);	// Count from 0 to 30,000 for a del	ay .				
4 5 P1OUT = RED	_OFF;	// Turn off the red LED light					
16 }							
8 }							
3							- PÍ
Console 33						1 I I I I I I I I I I I I I I I I I I I	
430: Loading complete.	There were 300 (co	de) and 16 (data) bytes written to FLA	5H/FRAM. The expected RAM	usage is 218 (uninitialized data + s	tack) bytes.		
				-			

58. The **Console** panel at the bottom will tell you how big your program is. In this example, the program was 300 + 16 = 316 bytes long.

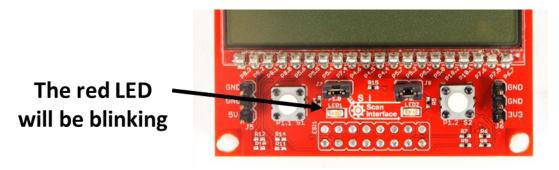




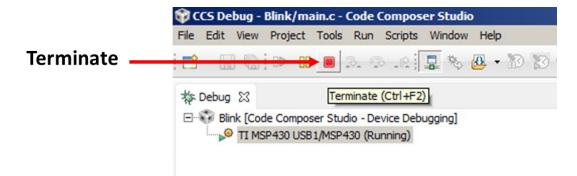
59. To start your program running, click the **Resume** (play) short-cut button on the toolbar.



60. Your red LED light should now be blinking! Congratulations, you have completed your first microcontroller program.

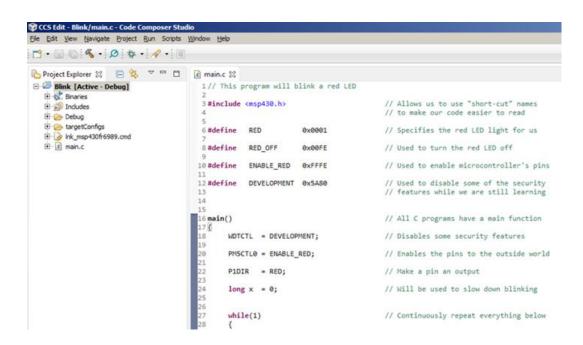


61. Let's leave the **CCS Debugger** and go back to the **Editor**. To do this, click on the **Terminate** short-cut button.





62. This closes the **Debugger** and opens the **Editor**.

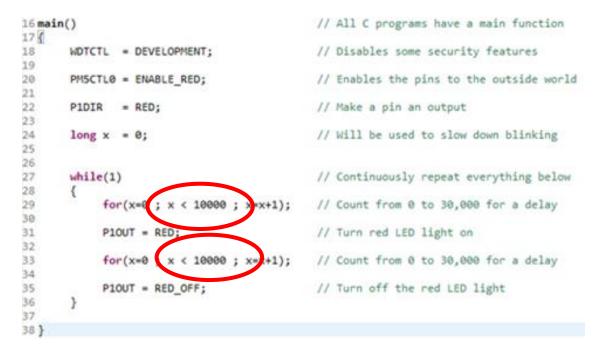


63. However, if you look at your Launchpad, you will see that the red LED is still blinking.

Terminate refers to terminating (ending) the **Debugger**. Your program will continue to run on the Launchpad.



64. Try changing the **30000** values in the **for** loops to implement different delay lengths. Try changing both values to **10000**. (Note, do not use 10,000 or 10.000 in **CCS**. Just **10000**.)



- 65. **Save** and **Build** your program. When complete, launch the **Debugger**.
- 66. Click **Resume** to start your new program. Notice that the red LED is blinking faster (3 times faster).
- 67. After trying out the **10000** value, you can **Terminate** the **Debugger** and try different values with the **Editor**. Just remember to **Save** and **Build** your program after you make any changes before launching the **Debugger**.
- 68. This concludes our Let's Get Started handout. Congratulations!