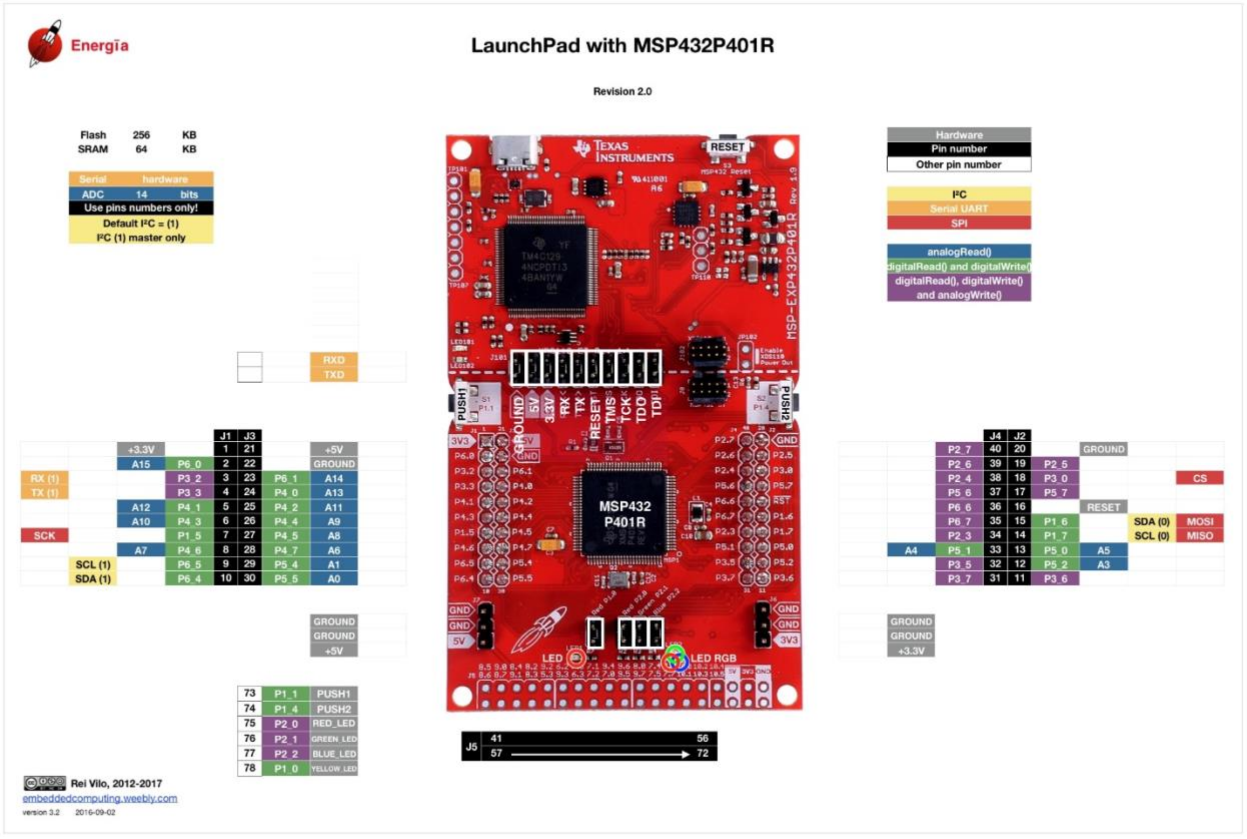
Introduction to Code Composer and the MSP432P401R

Introduction:

For this class you will be using a simple hardware development environment to introduce you to concepts associated with C Programming. You will use this environment as it allows more access to the details of the hardware. The objective of this lab is to allow you to understand how to use the TI Code Composer IDE.

Hardware:

The hardware you'll use for this lab is the MSP432P401R. Here is a picture of this hardware, with all the key parts of the hardware noted:



You'll use a USB cable to connect the hardware to your host computer.

Code Composer

In this first section of the lab you'll install Code Composer V10.

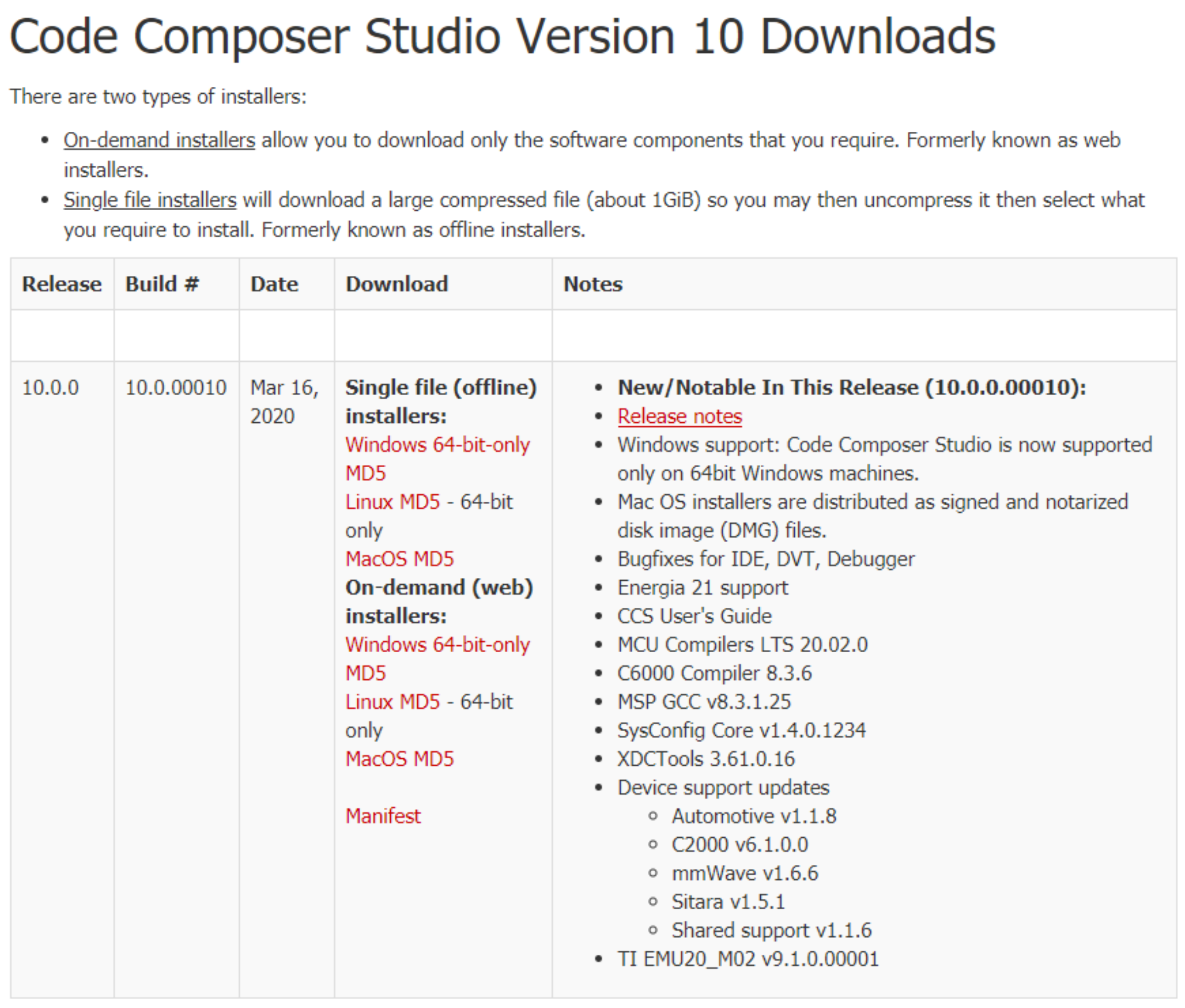
Installing the Software:

You'll have a choice in using Code Composer to develop you applications. You can either use the Cloud version, where the code actually is run on a remote server and you access the capabilities through a web browser, or you can use the downloadable version, where the code runs on your local machine. I prefer the latter, so I don't have to worry about always having internet access when I develop.

To install the software download the appropriate installer at:

https://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html

The selections will look like this:



Code Composer Studio Version 10 Downloads

There are two types of installers:

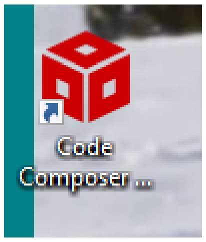
- On-demand installers allow you to download only the software components that you require. Formerly known as web installers.
- Single file installers will download a large compressed file (about 1GiB) so you may then uncompress it then select what you require to install. Formerly known as offline installers.

Release	Build #	Date	Download	Notes
10.0.0	10.0.00010	Mar 16, 2020	**Single file (offline) installers:** Windows 64-bit-only MD5 Linux MD5 - 64-bit only MacOS MD5 **On-demand (web) installers:** Windows 64-bit-only MD5 Linux MD5 - 64-bit only MacOS MD5 Manifest	• **New/Notable In This Release (10.0.0.00010):** • Release notes • Windows support: Code Composer Studio is now supported only on 64bit Windows machines. • Mac OS installers are distributed as signed and notarized disk image (DMG) files. • Bugfixes for IDE, DVT, Debugger • Energia 21 support • CCS User's Guide • MCU Compilers LTS 20.02.0 • C6000 Compiler 8.3.6 • MSP GCC v8.3.1.25 • SysConfig Core v1.4.0.1234 • XDCTools 3.61.0.16 • Device support updates ◦ Automotive v1.1.8 ◦ C2000 v6.1.0.0 ◦ mmWave v1.6.6 ◦ Sitara v1.5.1 ◦ Shared support v1.1.6 • TI EMU20_M02 v9.1.0.00001

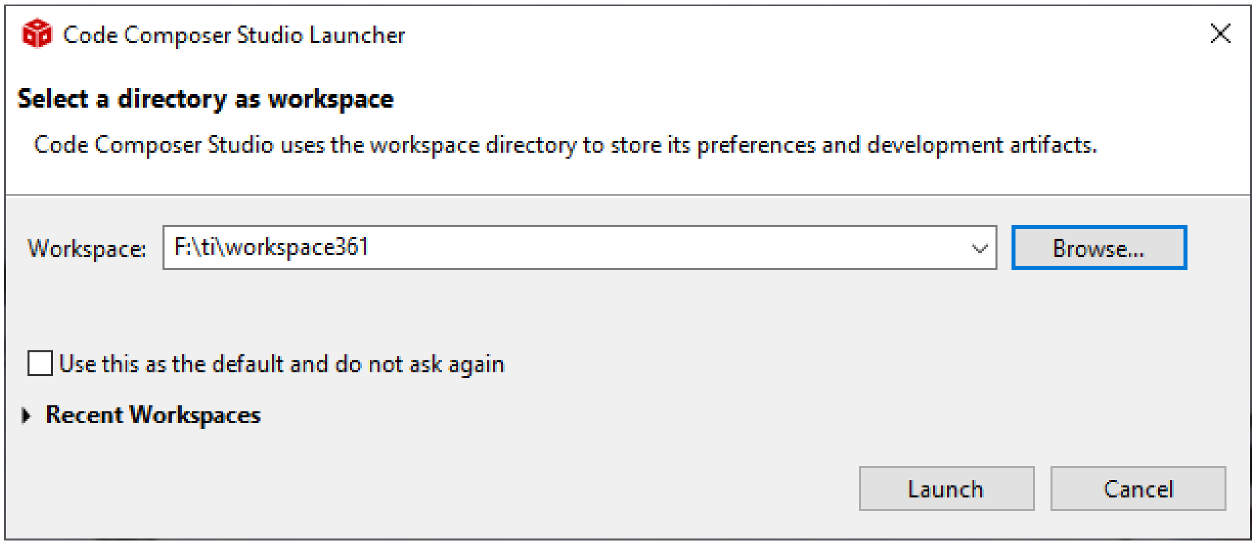
If will generally chose the On-demand (web) installers. Once you've installed you can then create your first application.

Creating your First Application:

Now that you have access to the software, let's create your very first application. First, start the software by double clicking on the icon:



As the program starts you should be prompted for the workspace you want to use.


Code Composer Studio Launcher

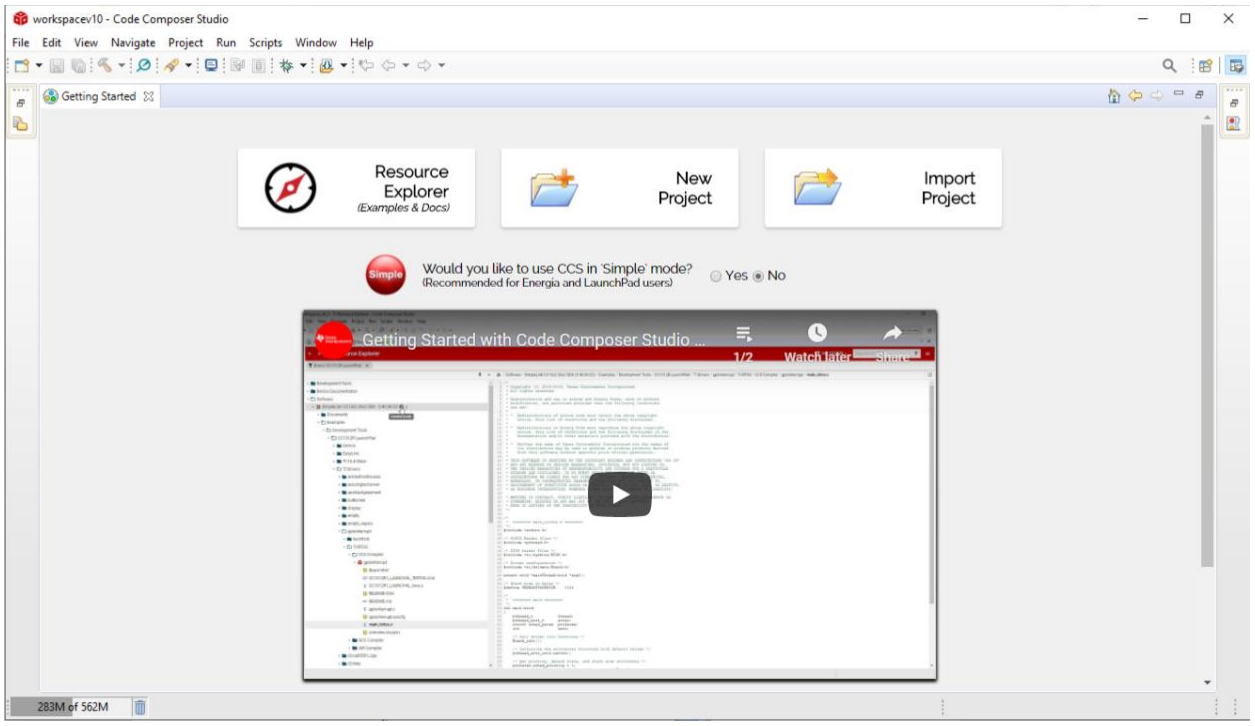
Select a directory as workspace

Code Composer Studio uses the workspace directory to store its preferences and development artifacts.

Workspace: F:\ti\workspace361

☐ Use this as the default and do not ask again

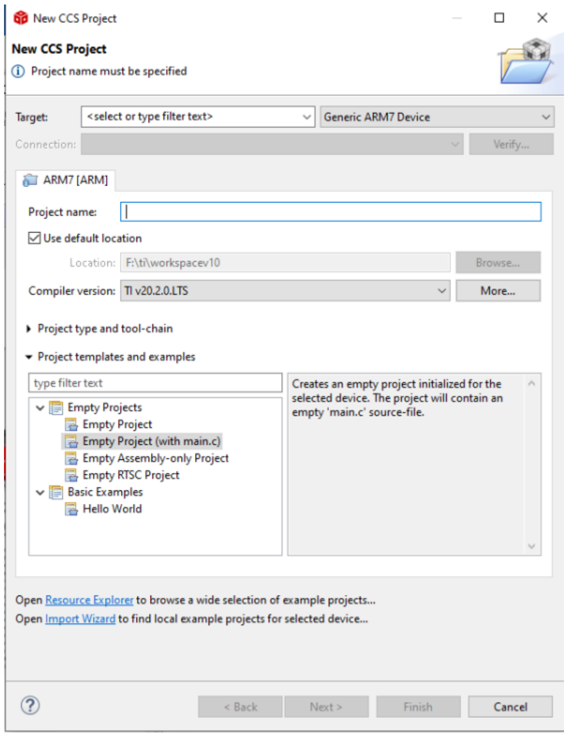
This workspace is the directory where you project files will be placed. You can use several different workspaces if you want to work on very different projects. Click Launch when you have specified the work space directory (you can just use the default.)


workspacev10 - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Getting Started
Resource Explorer (Examples & Docs)
New Project
Import Project
Would you like to use CCS in 'Simple' mode? (Recommended for Energia and LaunchPad users)
Yes [✓ No]
Getting Started with Code Composer Studio ...
283M of 562M

This is the introductory screen. Feel free to watch the Getting Started with Code Composer Studio v10.

If you haven't already connect your MSP432P401R to the computer via a USB cable.

To start a new project press the new Project button at the top of the screen. You should now see this selection:


New CCS Project

New CCS Project

Project name must be specified

Target: <select or type filter text> Generic ARM7 Device

Connection:

ARM7 [ARM]

Project name:

☑ Use default location

Location: F:\ti\workspacev10

Compiler version: TI v20.2.0.LTS

Project type and tool-chain

Project templates and examples

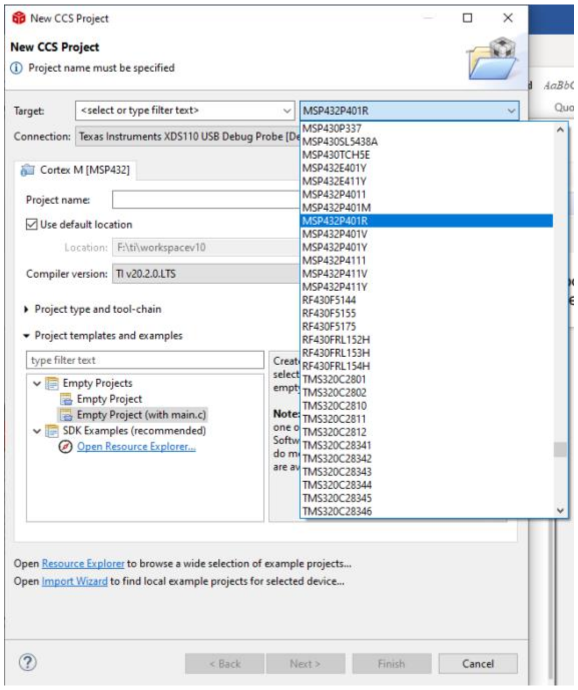
- Empty Projects
  - Empty Project
  - Empty Project (with main.c)
  - Empty Assembly-only Project
  - Empty RTSC Project
- Basic Examples
  - Hello World

Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Open Resource Explorer to browse a wide selection of example projects...

Open Import Wizard to find local example projects for selected device...

The first you'll want to do is the change the Target device. Under the Target: selection choose the MSP432P401R selection, like this:


New CCS Project

Project name must be specified

Target: <select or type filter text> MSP432P401R

Connection: Texas Instruments XDS110 USB Debug Probe [De

Cortex M [MSP432]

Project name:

Use default location

Location: F:\ti\workspacev10

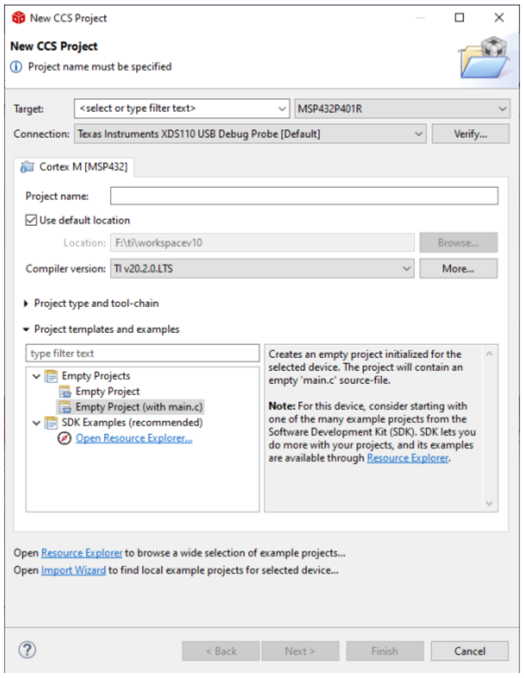
Compiler version: TI v20.2.0.LTS

MSP430P337
MSP430SL5438A
MSP430TCH5E
MSP432E401Y
MSP432E411Y
MSP432P4011
MSP432P401M
MSP432P401R
MSP432P401V
MSP432P401Y
MSP432P4111
MSP432P411V
MSP432P411Y
RF430F5144
RF430F5155
RF430F5175
RF430FRL152H
RF430FRL153H
RF430FRL154H
TMS320C2801
TMS320C2802
TMS320C2810
TMS320C2811
TMS320C2812
TMS320C28341
TMS320C28342
TMS320C28343
TMS320C28344
TMS320C28345
TMS320C28346

Open Resource Explorer to browse a wide selection of example projects...
Open Import Wizard to find local example projects for selected device...

Project type and tool-chain
Project templates and examples
type filter text
Empty Projects
Empty Project
Empty Project (with main.c)
SDK Examples (recommended)
Open Resource Explorer...
< Back
Next >
Finish
Cancel

You should now see this:


New CCS Project

Project name must be specified

Target: <select or type filter text> MSP432P401R

Connection: Texas Instruments XDS110 USB Debug Probe [Default]

Cortex M [MSP432]

Project name:

Use default location

Location: F:\ti\workspacev10

Compiler version: TI v20.2.0.LTS

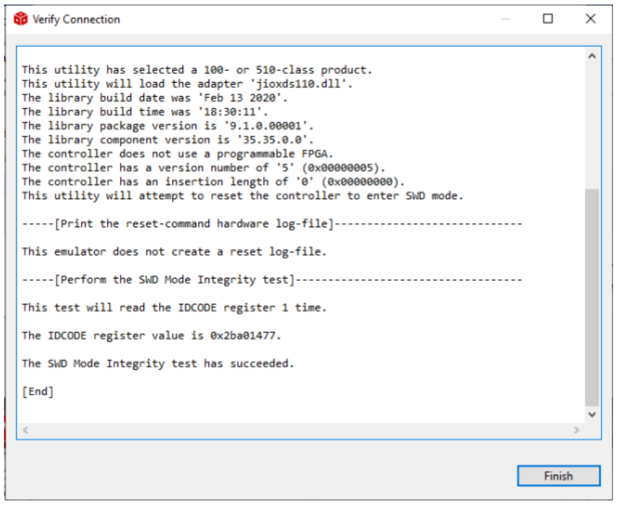
Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Note: For this device, consider starting with one of the many example projects from the Software Development Kit (SDK). SDK lets you do more with your projects, and its examples are available through Resource Explorer.

Open Resource Explorer to browse a wide selection of example projects...
Open Import Wizard to find local example projects for selected device...

Verify...
Browse...
More...
Project type and tool-chain
Project templates and examples
type filter text
Empty Projects
Empty Project
Empty Project (with main.c)
SDK Examples (recommended)
Open Resource Explorer...
< Back
Next >
Finish
Cancel

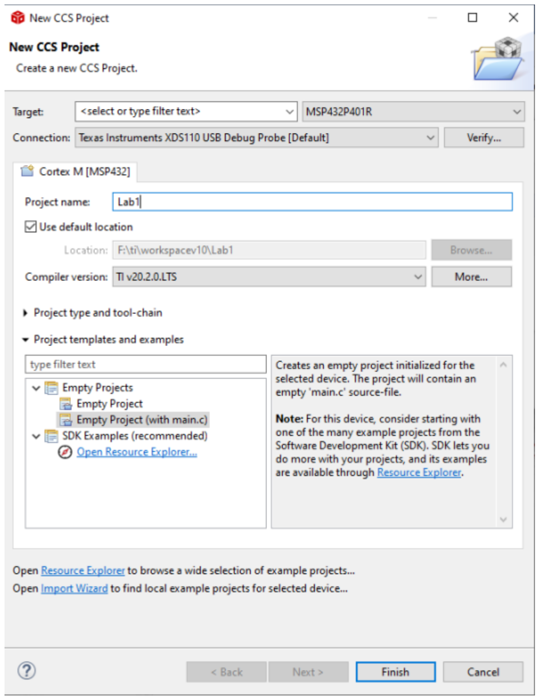
The Connection is showing the Default connection to this particular board, in this case it is a USB connection via the XDS110 interface (hardware on the board.) If you want to verify that Code Composer is connected to your board, then click Verify… and you should see this:



```
This utility has selected a 100- or 510-class product.
This utility will load the adapter 'jioxds110.dll'.
The library build date was 'Feb 13 2020'.
The library build time was '18:30:11'.
The library package version is '9.1.0.00001'.
The library component version is '35.35.0.0'.
The controller does not use a programmable FPGA.
The controller has a version number of '5' (0x00000005).
The controller has an insertion length of '0' (0x00000000).
This utility will attempt to reset the controller to enter SWD mode.

-----[Print the reset-command hardware log-file]-----------------------------

This emulator does not create a reset log-file.

-----[Perform the SWD Mode Integrity test]-----------------------------------

This test will read the IDCODE register 1 time.

The IDCODE register value is 0x2ba01477.

The SWD Mode Integrity test has succeeded.

[End]
```

Your device is connected, so click Finish.

Now you have a choice. You can create a blank project, a blank project with just main.c in it, or you can use one of the example projects. Let’s create a blank project with just main.c in it for a moment, just to look at the files that creates. So go ahead and type in a Project name and select Empty Project (with main.c) and select Finish.


New CCS Project
Verify...
Cortex M [MSP432]
Browse...
More...
Project type and tool-chain
Project templates and examples
Empty Projects
Empty Project
Empty Project (with main.c)
SDK Examples (recommended)
Open Resource Explorer...
< Back
Next >
Finish
Cancel

New CCS Project
Create a new CCS Project.

Target: <select or type filter text> MSP432P401R
Connection: Texas Instruments XDS110 USB Debug Probe [Default]

Project name: Lab1
☑ Use default location
Location: F:\ti\workspacev10\Lab1
Compiler version: TI v20.2.0.LTS

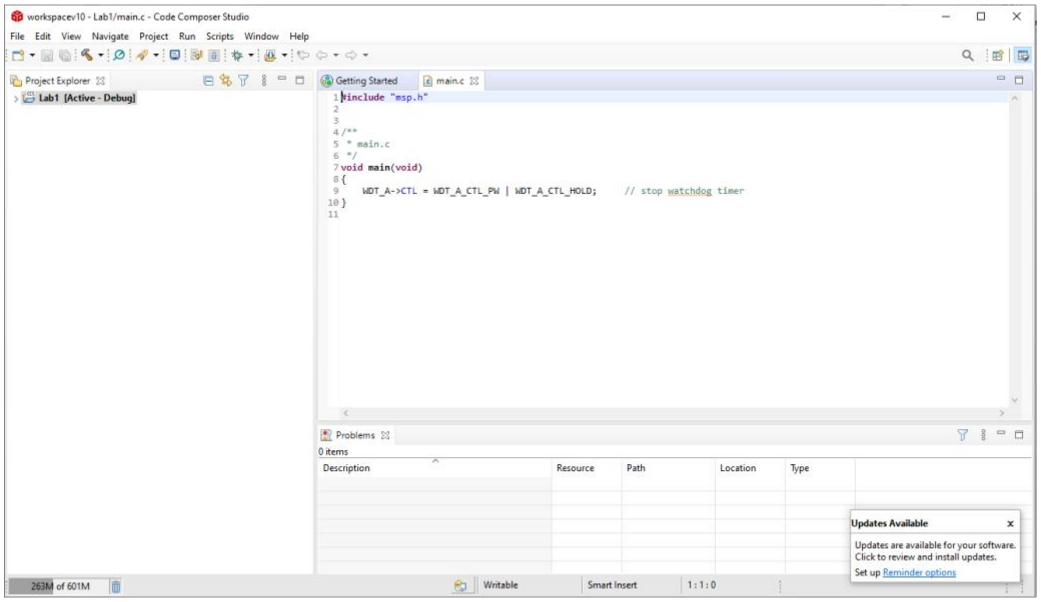
type filter text

Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Note: For this device, consider starting with one of the many example projects from the Software Development Kit (SDK). SDK lets you do more with your projects, and its examples are available through Resource Explorer.

Open Resource Explorer to browse a wide selection of example projects...
Open Import Wizard to find local example projects for selected device...

You should see this:



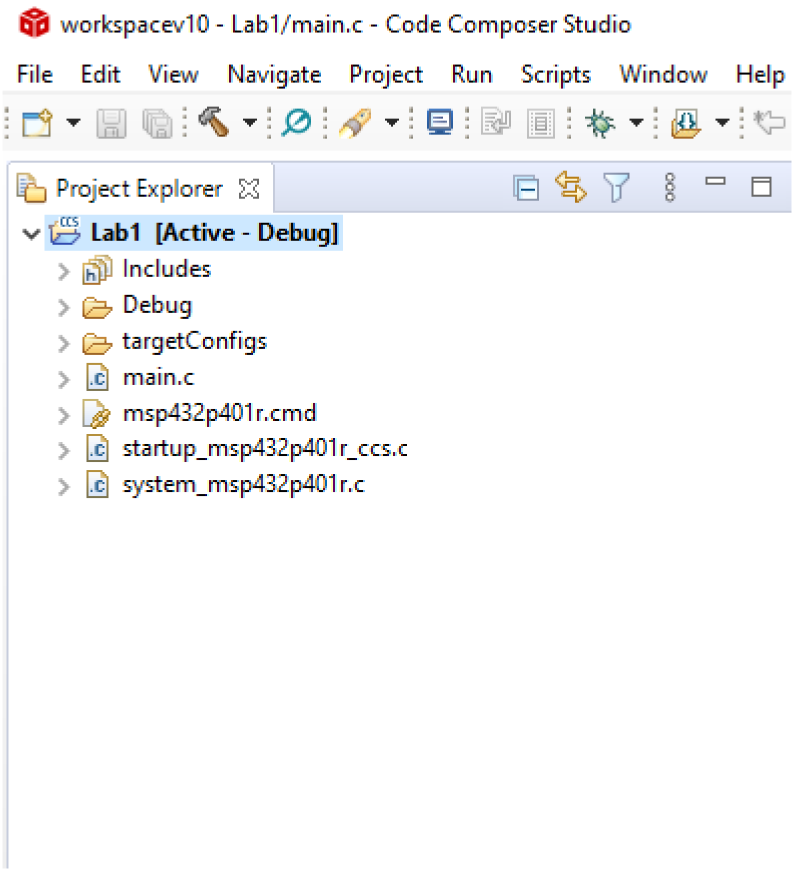
```
#include "msp.h"


/**
 * main.c
 */
void main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;     // stop watchdog timer
}

```

Updates Available

Updates are available for your software. Click to review and install updates.

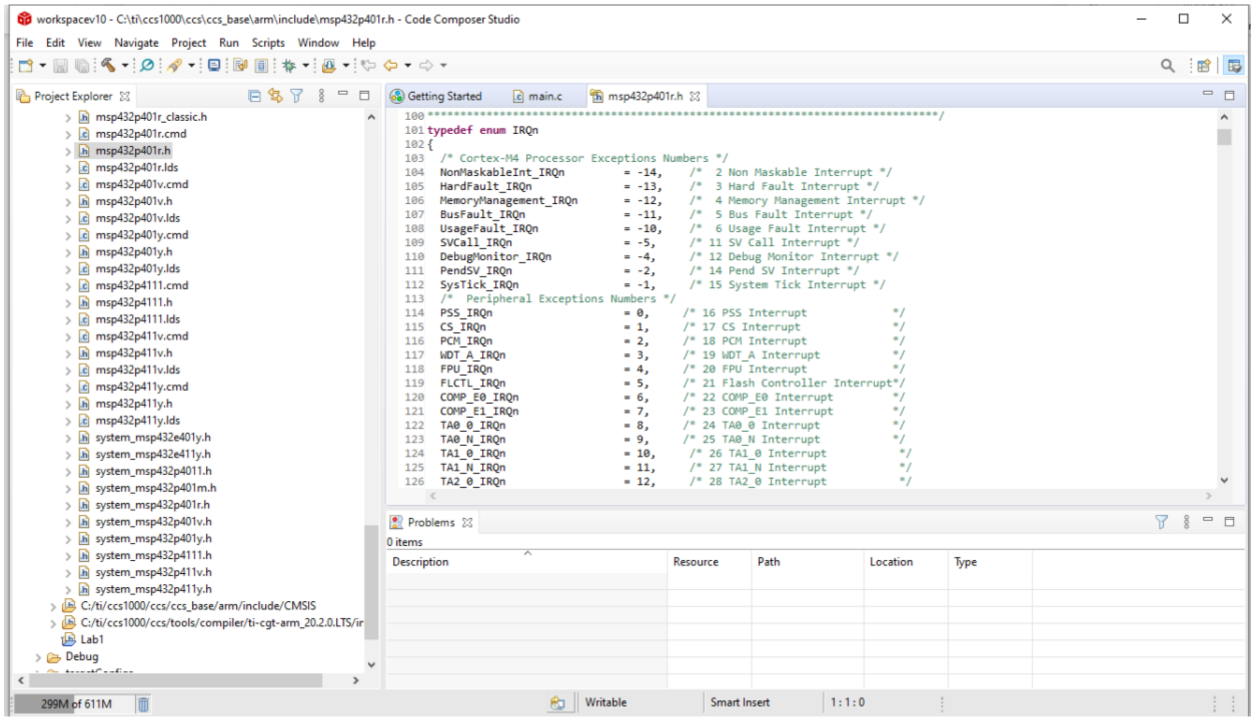
Set up Reminder options

Notice in the Upper Left you now have a blank project. The screen is also showing you your main.c file. To look at all the files that your action created, select the > just to the left of the Blank Project label. You should see something like this:


workspacev10 - Lab1/main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
Lab1 [Active - Debug]
Includes
Debug
targetConfigs
main.c
msp432p401r.cmd
startup_msp432p401r_ccs.c
system_msp432p401r.c

There are several directories and files that are created. Let's look at each briefly.

Includes – These files are the set of includes for all of the libraries that will be compiled together when you build your code. Let's look at just one of these files:



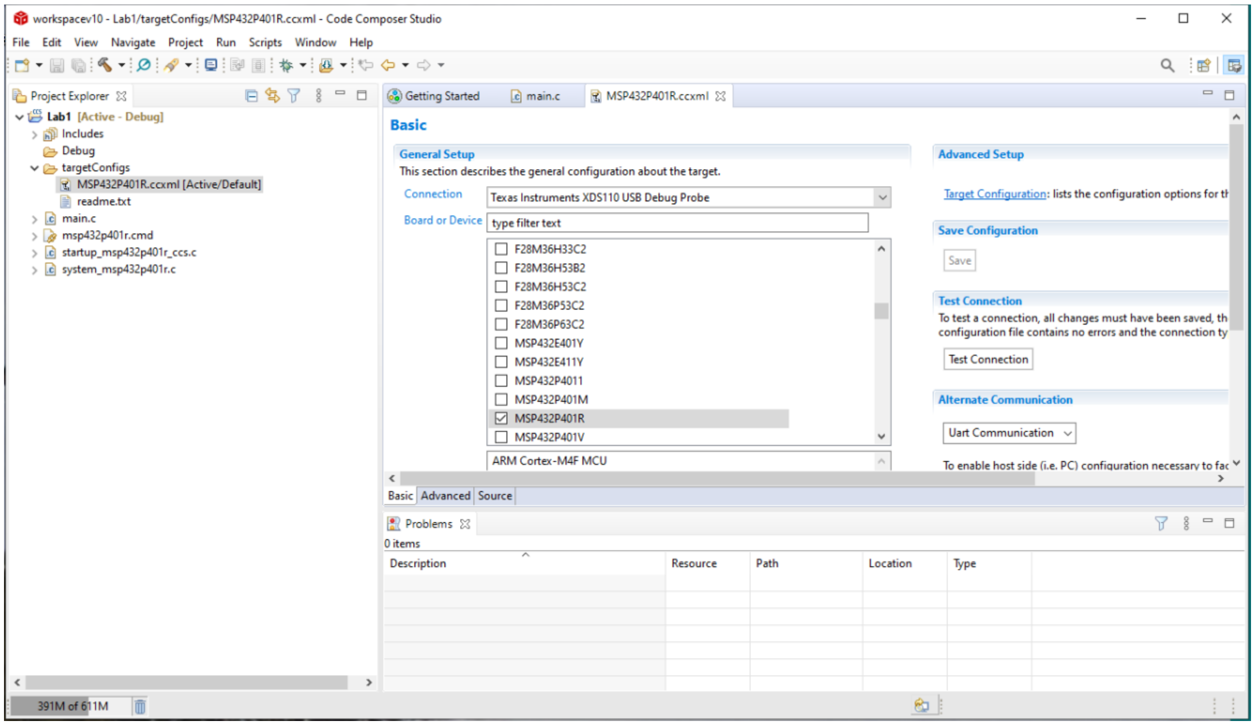
```
*****************************************************************************/
typedef enum IRQn
{
  /* Cortex-M4 Processor Exceptions Numbers */
  NonMaskableInt_IRQn         = -14,    /*  2 Non Maskable Interrupt */
  HardFault_IRQn              = -13,    /*  3 Hard Fault Interrupt */
  MemoryManagement_IRQn       = -12,    /*  4 Memory Management Interrupt */
  BusFault_IRQn               = -11,    /*  5 Bus Fault Interrupt */
  UsageFault_IRQn             = -10,    /*  6 Usage Fault Interrupt */
  SVCall_IRQn                 = -5,     /* 11 SV Call Interrupt */
  DebugMonitor_IRQn           = -4,     /* 12 Debug Monitor Interrupt */
  PendSV_IRQn                 = -2,     /* 14 Pend SV Interrupt */
  SysTick_IRQn                = -1,     /* 15 System Tick Interrupt */
  /*  Peripheral Exceptions Numbers */
  PSS_IRQn                    = 0,     /* 16 PSS Interrupt             */
  CS_IRQn                     = 1,     /* 17 CS Interrupt              */
  PCM_IRQn                    = 2,     /* 18 PCM Interrupt             */
  WDT_A_IRQn                  = 3,     /* 19 WDT_A Interrupt           */
  FPU_IRQn                    = 4,     /* 20 FPU Interrupt             */
  FLCTL_IRQn                  = 5,     /* 21 Flash Controller Interrupt*/
  COMP_E0_IRQn                = 6,     /* 22 COMP_E0 Interrupt         */
  COMP_E1_IRQn                = 7,     /* 23 COMP_E1 Interrupt         */
  TA0_0_IRQn                  = 8,     /* 24 TA0_0 Interrupt           */
  TA0_N_IRQn                  = 9,     /* 25 TA0_N Interrupt           */
  TA1_0_IRQn                  = 10,     /* 26 TA1_0 Interrupt           */
  TA1_N_IRQn                  = 11,     /* 27 TA1_N Interrupt           */
  TA2_0_IRQn                  = 12,     /* 28 TA2_0 Interrupt           */
```

In this case I've opened the C:/ti/ccsv8/ccs_base/arm/include directory by clicking on the arrow just to the left of it, then double clicking on the file msp432p401r.h file. This has a large set of enums (definitional structures that turn numbers into names in the code) that are declared to support this hardware. I would never edit this file, but if I wanted to know what a specific number is defined as in the hardware, I could go into this file.

Get rid of the msp432p401r.h file in the editor area by clicking on the x. And you can also un-expand the directory by clicking on the arrow key just to the left of it.

Debug – There is nothing in this file, as you haven't done any Debug. We'll cover this later.

targetConfigs – This directory specifies your hardware connection. If you select the MSP432P401R.ccxml file you will find an active page where you can set this. Don't change it.



workspacev10 - Lab1/targetConfigs/MSP432P401R.ccxml - Code Composer Studio

Basic

General Setup

This section describes the general configuration about the target.

Connection: Texas Instruments XDS110 USB Debug Probe

Board or Device: type filter text

- [ ] F28M36H33C2
- [ ] F28M36H53B2
- [ ] F28M36H53C2
- [ ] F28M36P53C2
- [ ] F28M36P63C2
- [ ] MSP432E401Y
- [ ] MSP432E411Y
- [ ] MSP432P4011
- [ ] MSP432P401M
- [x] MSP432P401R
- [ ] MSP432P401V

ARM Cortex-M4F MCU

Advanced Setup

Target Configuration: lists the configuration options for th

Save Configuration

Save

Test Connection

To test a connection, all changes must have been saved, th configuration file contains no errors and the connection ty

Test Connection

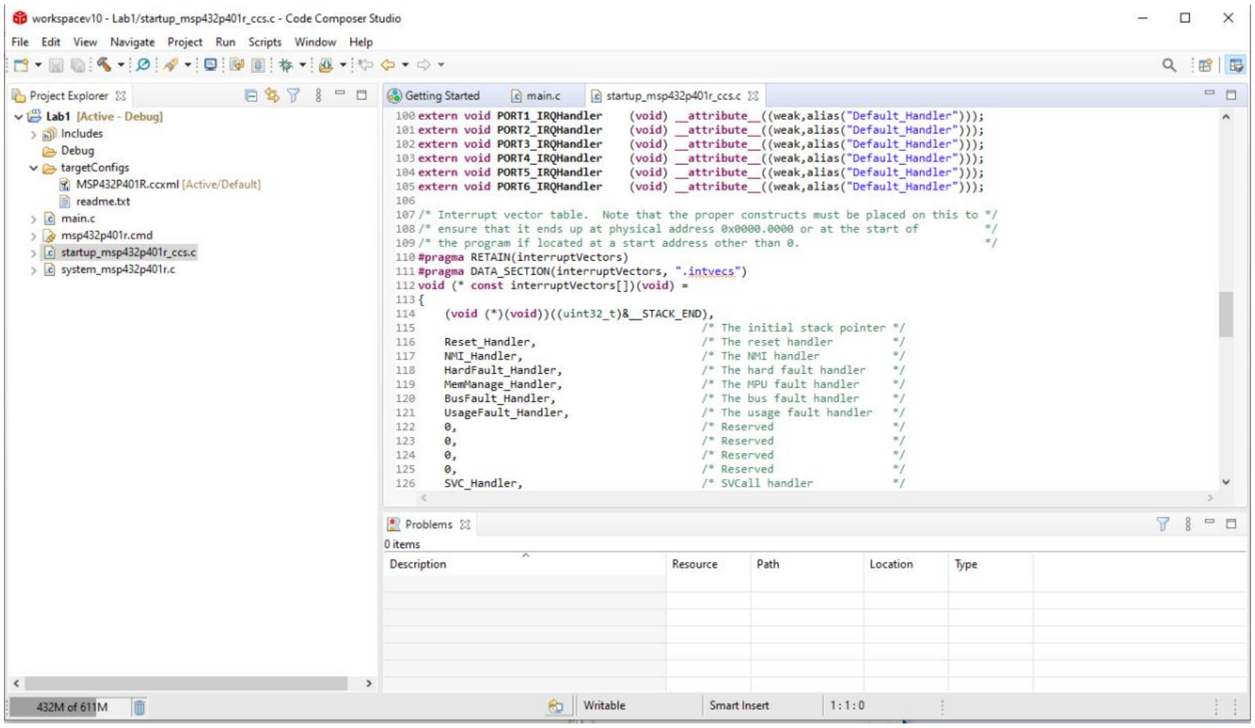
Alternate Communication

Uart Communication

To enable host side (i.e. PC) configuration necessary to fac

There are two other files in this directory that are code files. The main.c file, as you have already seen, holds the main() functions. Here are details about the other two files:

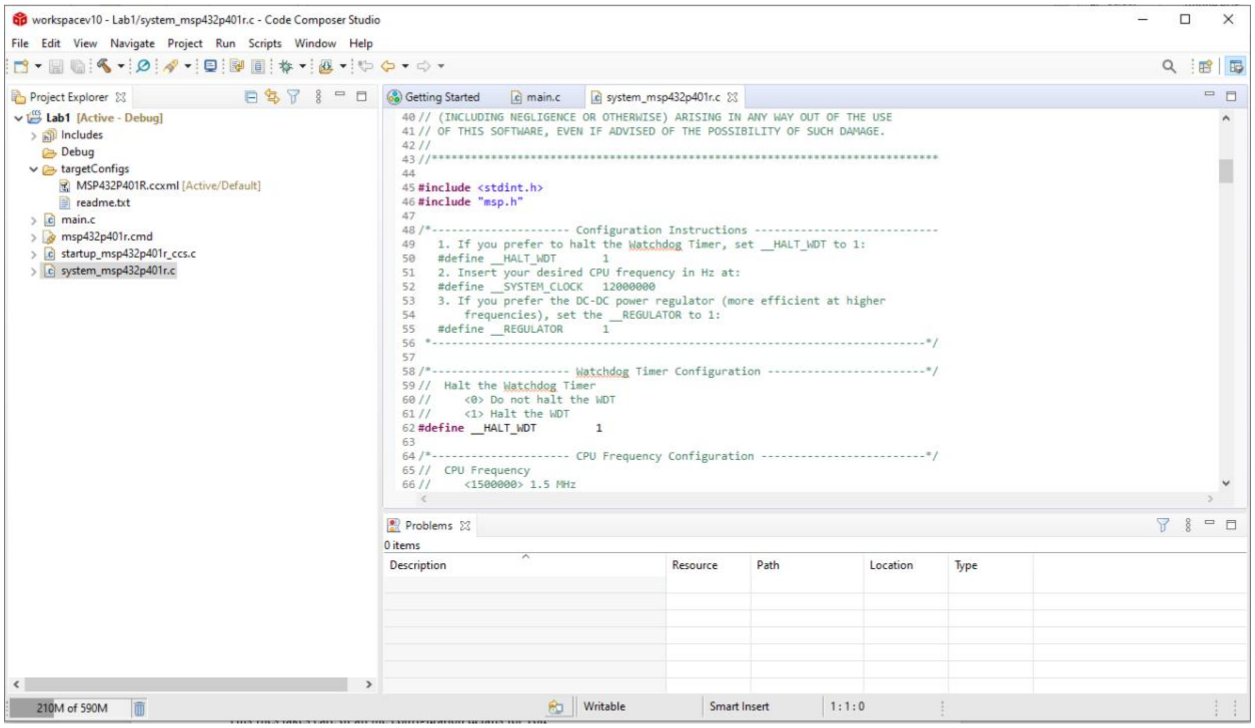
startup_msp432p401r_ccs.c – This holds the interrupt vector table. We will talk about his later in the class, but is specifies the functions to run when a specific interrupt occurs. Here is what that table looks like:



workspacev10 - Lab1/startup_msp432p401r_ccs.c - Code Composer Studio

```c
extern void PORT1_IRQHandler    (void) __attribute__((weak,alias("Default_Handler")));
extern void PORT2_IRQHandler    (void) __attribute__((weak,alias("Default_Handler")));
extern void PORT3_IRQHandler    (void) __attribute__((weak,alias("Default_Handler")));
extern void PORT4_IRQHandler    (void) __attribute__((weak,alias("Default_Handler")));
extern void PORT5_IRQHandler    (void) __attribute__((weak,alias("Default_Handler")));
extern void PORT6_IRQHandler    (void) __attribute__((weak,alias("Default_Handler")));

/* Interrupt vector table.  Note that the proper constructs must be placed on this to */
/* ensure that it ends up at physical address 0x0000.0000 or at the start of          */
/* the program if located at a start address other than 0.                           */
#pragma RETAIN(interruptVectors)
#pragma DATA_SECTION(interruptVectors, ".intvecs")
void (* const interruptVectors[])(void) =
{
    (void (*)(void))((uint32_t)&__STACK_END),
                                            /* The initial stack pointer */
    Reset_Handler,                          /* The reset handler         */
    NMI_Handler,                            /* The NMI handler           */
    HardFault_Handler,                      /* The hard fault handler    */
    MemManage_Handler,                      /* The MPU fault handler     */
    BusFault_Handler,                       /* The bus fault handler     */
    UsageFault_Handler,                     /* The usage fault handler   */
    0,                                      /* Reserved                  */
    0,                                      /* Reserved                  */
    0,                                      /* Reserved                  */
    0,                                      /* Reserved                  */
    SVC_Handler,                            /* SVCall handler            */
```

system_msp432p401r.c – This file sets up all sorts of configuration details. Here is a view of bit of the file:

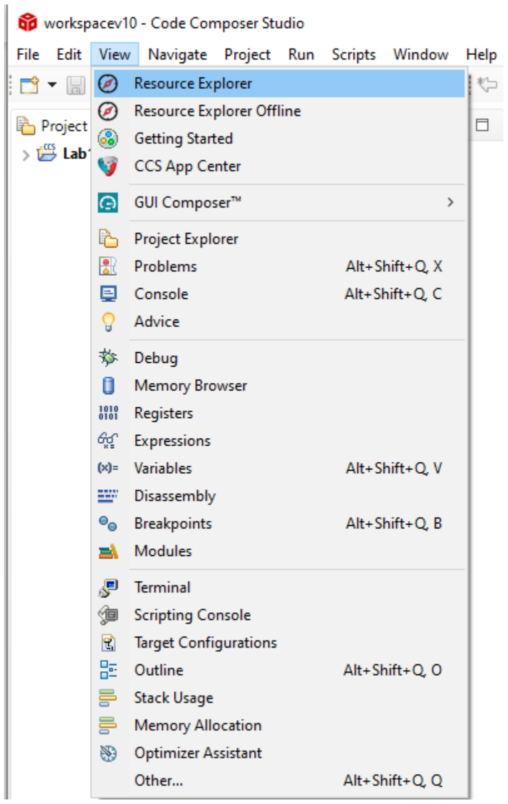


```
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//*****************************************************************************

#include <stdint.h>
#include "msp.h"

/*-------------------- Configuration Instructions ----------------------------
   1. If you prefer to halt the Watchdog Timer, set __HALT_WDT to 1:
   #define __HALT_WDT       1
   2. Insert your desired CPU frequency in Hz at:
   #define __SYSTEM_CLOCK   12000000
   3. If you prefer the DC-DC power regulator (more efficient at higher
       frequencies), set the __REGULATOR to 1:
   #define __REGULATOR      1
 *---------------------------------------------------------------------------*/

/*-------------------- Watchdog Timer Configuration -----------------------*/
//  Halt the Watchdog Timer
//     <0> Do not halt the WDT
//     <1> Halt the WDT
#define __HALT_WDT         1

/*-------------------- CPU Frequency Configuration ------------------------*/
//  CPU Frequency
//     <1500000> 1.5 MHz
```

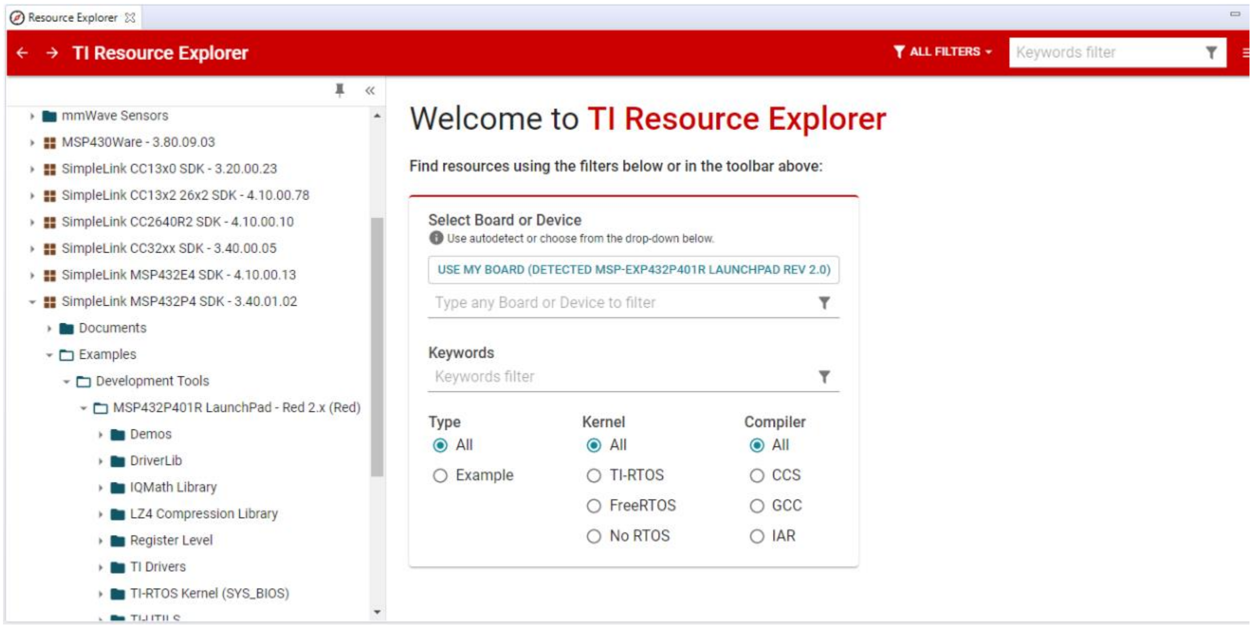
This files takes care of all the configuration details for you.

Now if you were going to add your code, you’d start by adding code to main.c in the main() function. But instead, let’s start with a file that has a bit more framework to start with. So click the New Project selection from the main screen (to get back to it just close all the other files in the large main window by clicking the x next to them.)

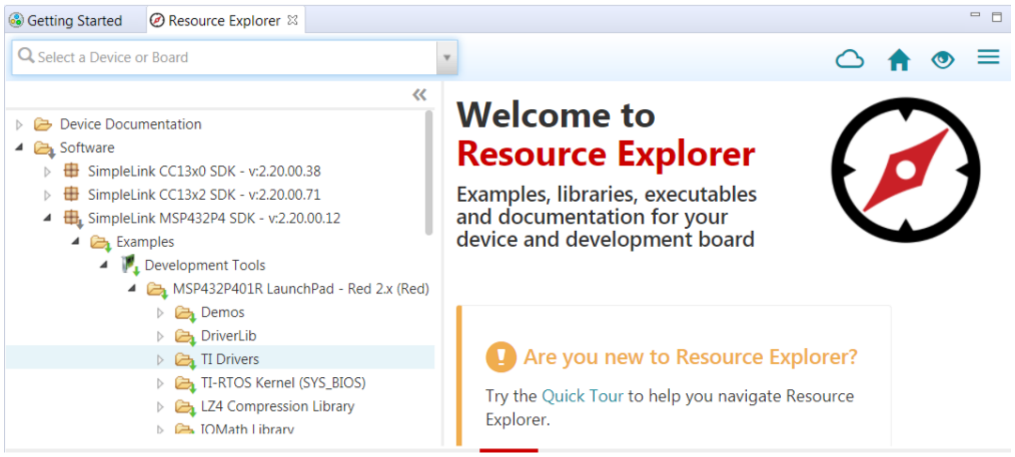
This time go to the View selection and chose the Resource Explorer…



The following should pop up:

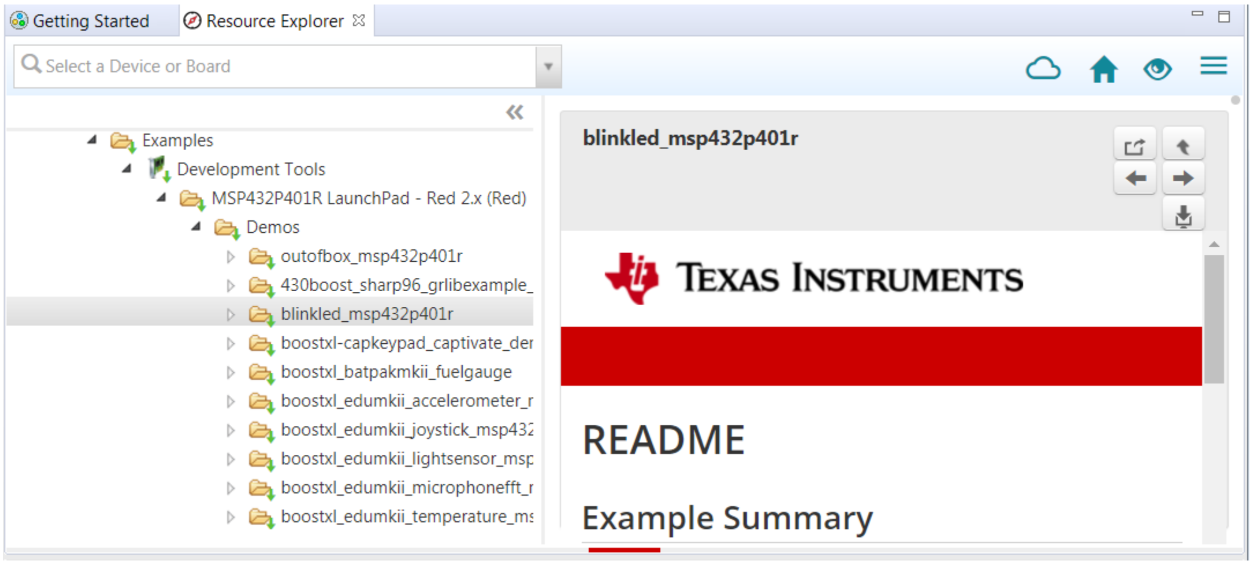

Resource Explorer
TI Resource Explorer
ALL FILTERS
Keywords filter
mmWave Sensors
MSP430Ware - 3.80.09.03
SimpleLink CC13x0 SDK - 3.20.00.23
SimpleLink CC13x2 26x2 SDK - 4.10.00.78
SimpleLink CC2640R2 SDK - 4.10.00.10
SimpleLink CC32xx SDK - 3.40.00.05
SimpleLink MSP432E4 SDK - 4.10.00.13
SimpleLink MSP432P4 SDK - 3.40.01.02
Documents
Examples
Development Tools
MSP432P401R LaunchPad - Red 2.x (Red)
Demos
DriverLib
IQMath Library
LZ4 Compression Library
Register Level
TI Drivers
TI-RTOS Kernel (SYS_BIOS)
Welcome to TI Resource Explorer
Find resources using the filters below or in the toolbar above:
Select Board or Device
Use autodetect or choose from the drop-down below.
USE MY BOARD (DETECTED MSP-EXP432P401R LAUNCHPAD REV 2.0)
Type any Board or Device to filter
Keywords
Keywords filter
Type
All
Example
Kernel
All
TI-RTOS
FreeRTOS
No RTOS
Compiler
All
CCS
GCC
IAR

You will need to be connected to the Internet to see this. Now the only challenge with using the Resource Explorer is the large choice of possible examples. Let's get to those that apply to your board:

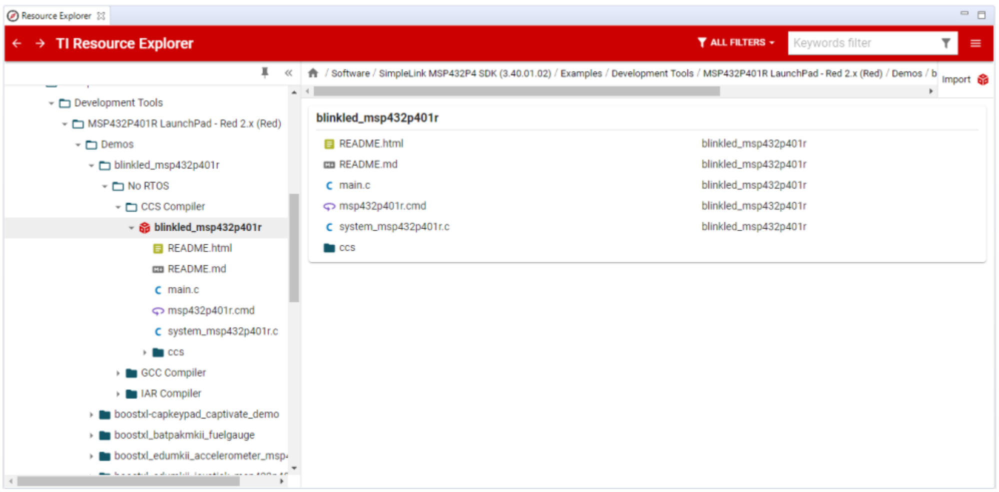

Getting Started
Resource Explorer
Select a Device or Board
Device Documentation
Software
SimpleLink CC13x0 SDK - v:2.20.00.38
SimpleLink CC13x2 SDK - v:2.20.00.71
SimpleLink MSP432P4 SDK - v:2.20.00.12
Examples
Development Tools
MSP432P401R LaunchPad - Red 2.x (Red)
Demos
DriverLib
TI Drivers
TI-RTOS Kernel (SYS_BIOS)
LZ4 Compression Library
IQMath Library
Welcome to
Resource Explorer
Examples, libraries, executables and documentation for your device and development board
Are you new to Resource Explorer?
Try the Quick Tour to help you navigate Resource Explorer.

Expand the directories by clicking on the arrow at the left of each directory. You are now at the MSP432P401R LaunchPad – Red set. There are still many examples here, so it will take some understanding to figure out where to select your files from.

Each of these directories represent not only a set of example files, but also driver files that abstract, or hide, some of the low level hardware access via functions. In this example you'll focus on the Demos directory, so select that and you should see these choices:


Getting Started
Resource Explorer
Select a Device or Board
Examples
Development Tools
MSP432P401R LaunchPad - Red 2.x (Red)
Demos
outofbox_msp432p401r
430boost_sharp96_grlibexample_
blinkled_msp432p401r
boostxl-capkeypad_captivate_der
boostxl_batpakmkii_fuelgauge
boostxl_edumkii_accelerometer_r
boostxl_edumkii_joystick_msp432
boostxl_edumkii_lightsensor_msp
boostxl_edumkii_microphonefft_r
boostxl_edumkii_temperature_ms
blinkled_msp432p401r
Texas Instruments
README
Example Summary

Each of these are a project, although some/many will need additional booster packs (hardware shields) to provide their functionality. Now select blinkled_msp432p401r->No RTOS->CCS Compiler->blinkled_msp432p401r from the list.

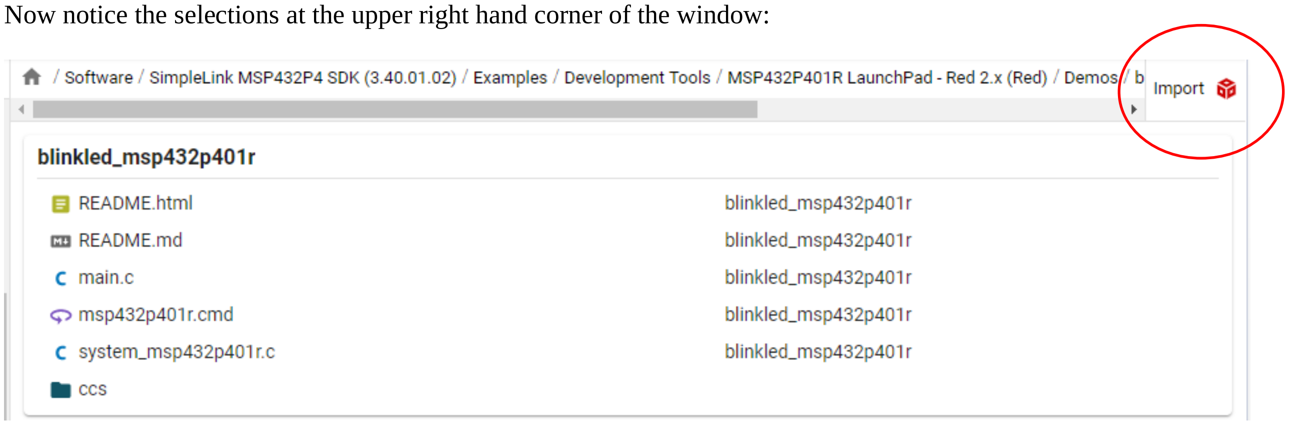


/ Software / SimpleLink MSP432P4 SDK (3.40.01.02) / Examples / Development Tools / MSP432P401R LaunchPad - Red 2.x (Red) / Demos / b

blinkled_msp432p401r

README.html	blinkled_msp432p401r
README.md	blinkled_msp432p401r
main.c	blinkled_msp432p401r
msp432p401r.cmd	blinkled_msp432p401r
system_msp432p401r.c	blinkled_msp432p401r
ccs	

Now notice the selections at the upper right hand corner of the window:

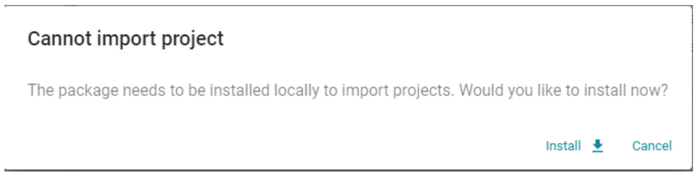


/ Software / SimpleLink MSP432P4 SDK (3.40.01.02) / Examples / Development Tools / MSP432P401R LaunchPad - Red 2.x (Red) / Demos / b

blinkled_msp432p401r

README.html	blinkled_msp432p401r
README.md	blinkled_msp432p401r
main.c	blinkled_msp432p401r
msp432p401r.cmd	blinkled_msp432p401r
system_msp432p401r.c	blinkled_msp432p401r
ccs	

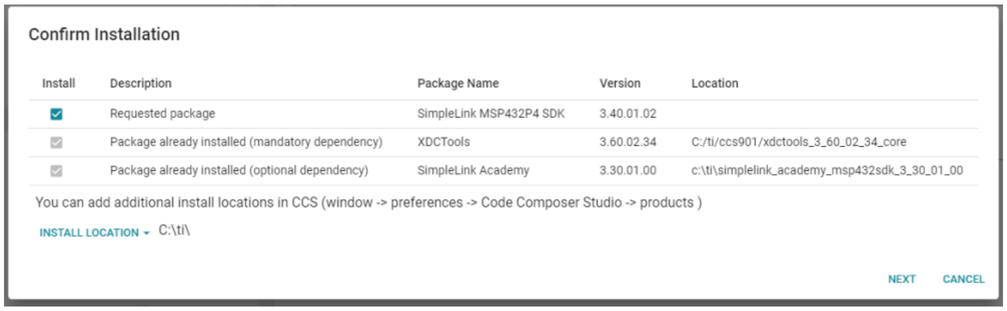
You are going to select the import selection. You should see this selection:



Cannot import project

The package needs to be installed locally to import projects. Would you like to install now?

Now click Install. Then you should see this:



Confirm Installation

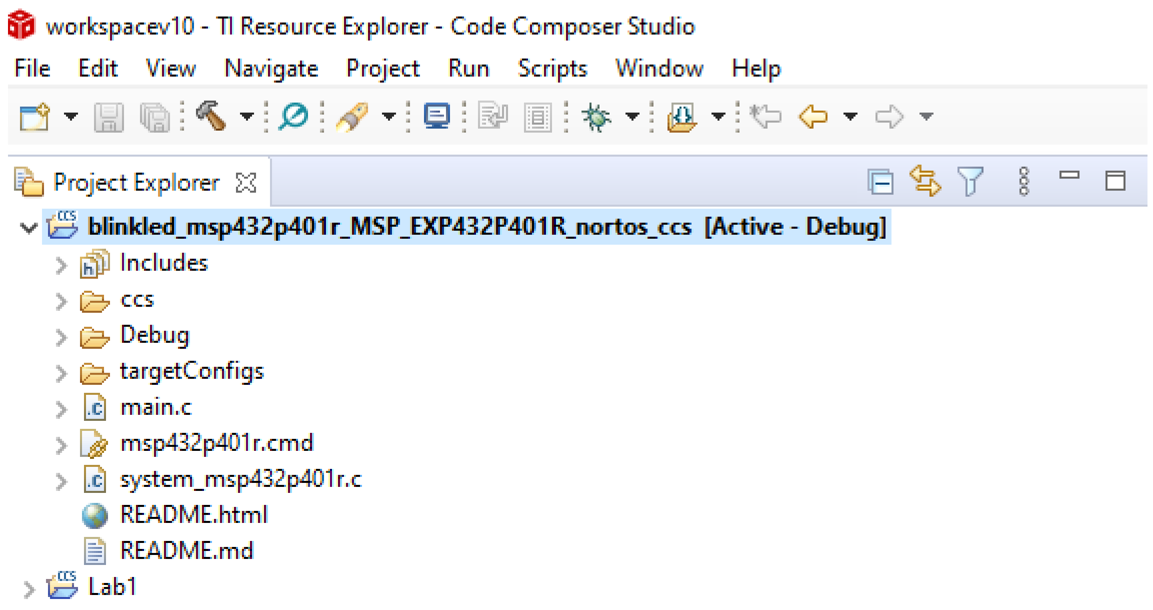
Install	Description	Package Name	Version	Location
✓	Requested package	SimpleLink MSP432P4 SDK	3.40.01.02	
✓	Package already installed (mandatory dependency)	XDCTools	3.60.02.34	C:/ti/ccs901/xdctools_3_60_02_34_core
✓	Package already installed (optional dependency)	SimpleLink Academy	3.30.01.00	c:\ti\simplelink_academy_msp432sdk_3_30_01_00

You can add additional install locations in CCS (window -> preferences -> Code Composer Studio -> products)

INSTALL LOCATION C:\ti\

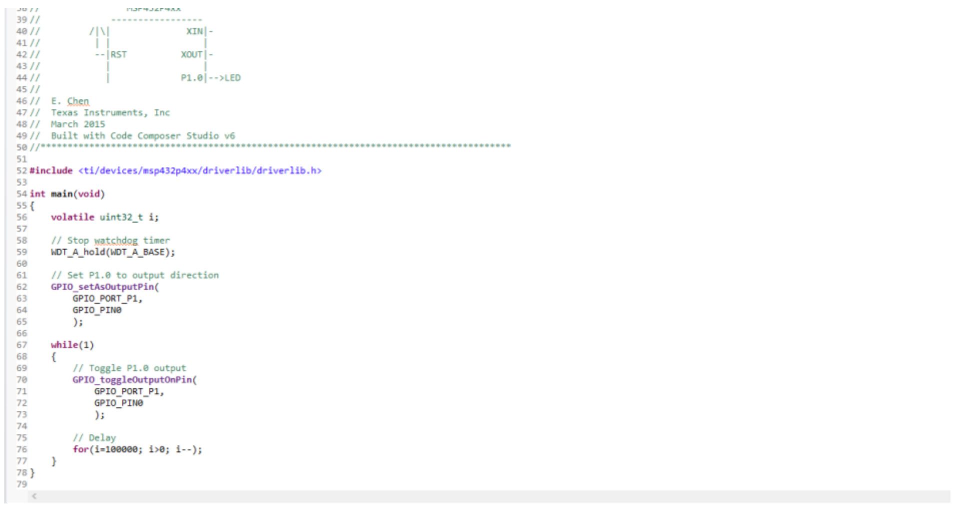
Click NEXT. Then OK on the next dialogue box. It might take some time, but you'll see the final dialogue box to confirm the import of the project.

You should then see the project in your project window on the left (I've expanded the directory so that you can see all the files.)



```
blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs  [Active - Debug]
    Includes
    ccs
    Debug
    targetConfigs
    main.c
    msp432p401r.cmd
    system_msp432p401r.c
    README.html
    README.md
Lab1
```

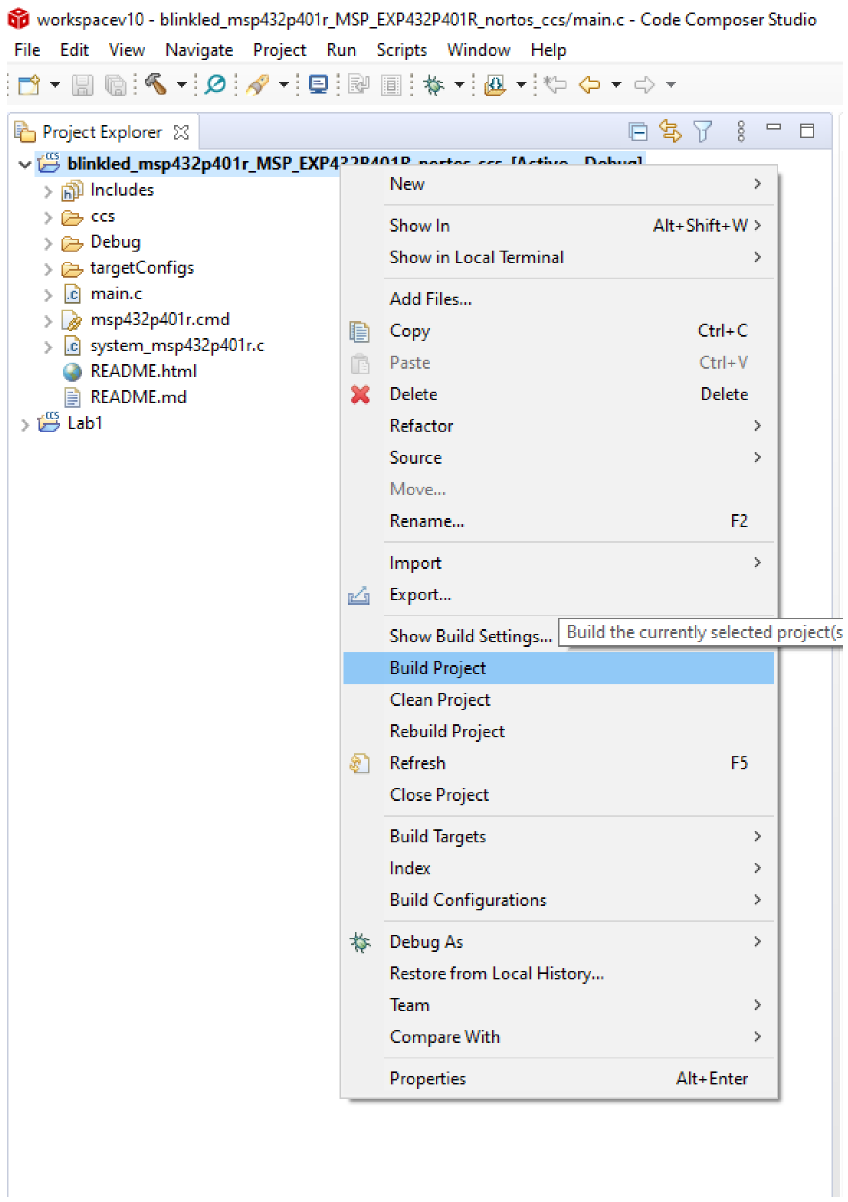
Now close the Resource tab in the main window, and double click on the main.c. You should see the code:



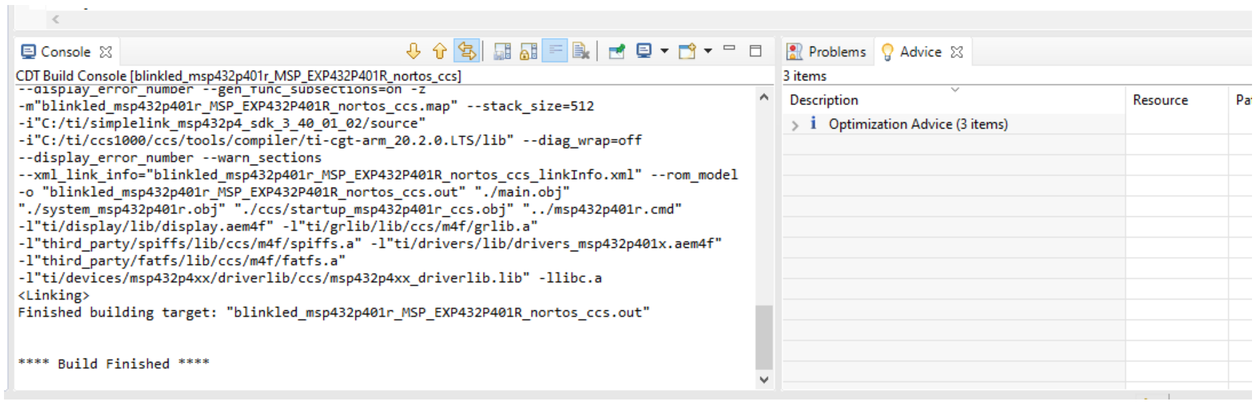
```
//            -----------------
//        /|\|              XIN|-
//         | |                 |
//         --|RST          XOUT|-
//           |                 |
//           |             P1.0|-->LED
//
//   E. Chen
//   Texas Instruments, Inc
//   March 2015
//   Built with Code Composer Studio v6
//******************************************************************************

#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

int main(void)
{
    volatile uint32_t i;

    // Stop watchdog timer
    WDT_A_hold(WDT_A_BASE);

    // Set P1.0 to output direction
    GPIO_setAsOutputPin(
        GPIO_PORT_P1,
        GPIO_PIN0
        );

    while(1)
    {
        // Toggle P1.0 output
        GPIO_toggleOutputOnPin(
            GPIO_PORT_P1,
            GPIO_PIN0
            );

        // Delay
        for(i=100000; i>0; i--);
    }
}
```

Now important to note is the inclusion of the library <ti/devices/msp432p4xx/driverlib/driverlib.h> . This provides some hardware abstraction (hiding of the low level hardware) by providing functions like GPIO-setAsOutputPin and GPIO_toggleOutputOnPin. You can call these functions without knowing the details of what is going on at the lower levels.

Now that you have this project, if you want to run it on your board you need to first compile it. To do this right click on the project name. You should see this selection:

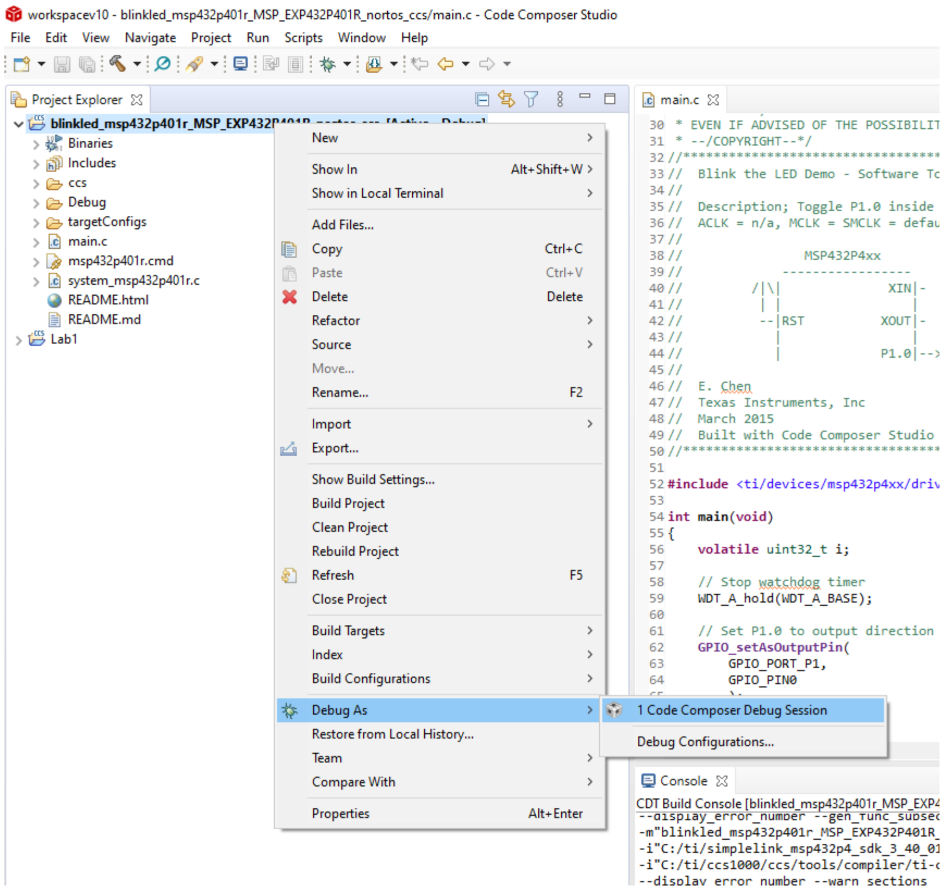


Now select the Build Project from this list. If everything went well you should see this:



```
CDT Build Console [blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs]
--display_error_number --gen_func_subsections=on -z
-m"blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs.map" --stack_size=512
-i"C:/ti/simplelink_msp432p4_sdk_3_40_01_02/source"
-i"C:/ti/ccs1000/ccs/tools/compiler/ti-cgt-arm_20.2.0.LTS/lib" --diag_wrap=off
--display_error_number --warn_sections
--xml_link_info="blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs_linkInfo.xml" --rom_model
-o "blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs.out" "./main.obj"
"./system_msp432p401r.obj" "./ccs/startup_msp432p401r_ccs.obj" "../msp432p401r.cmd"
-l"ti/display/lib/display.aem4f" -l"ti/grlib/lib/ccs/m4f/grlib.a"
-l"third_party/spiffs/lib/ccs/m4f/spiffs.a" -l"ti/drivers/lib/drivers_msp432p401x.aem4f"
-l"third_party/fatfs/lib/ccs/m4f/fatfs.a"
-l"ti/devices/msp432p4xx/driverlib/ccs/msp432p4xx_driverlib.lib" -llibc.a
<Linking>
Finished building target: "blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs.out"


**** Build Finished ****
```

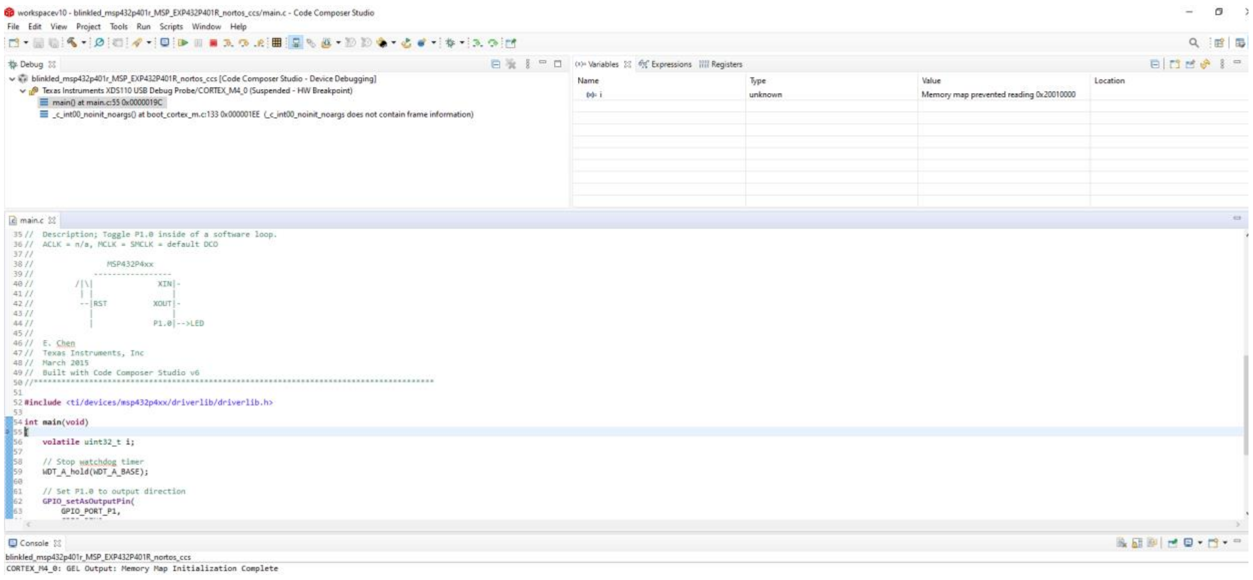
In the CDT Build Console window you should see the output of the Build process. Now that the code is ready let's send it over to our board. To do this right click on the Project name again. This time choose the Debug As->Code Composer Debug Session like this:



```
 * EVEN IF ADVISED OF THE POSSIBILIT
 * --/COPYRIGHT--*/
//***********************************
//  Blink the LED Demo - Software To
//
//  Description; Toggle P1.0 inside
//  ACLK = n/a, MCLK = SMCLK = defau
//
//                MSP432P4xx
//             -----------------
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|-
//            |                 |
//            |             P1.0|-->
//
//  E. Chen
//  Texas Instruments, Inc
//  March 2015
//  Built with Code Composer Studio
//***********************************

#include <ti/devices/msp432p4xx/driv

int main(void)
{
    volatile uint32_t i;

    // Stop watchdog timer
    WDT_A_hold(WDT_A_BASE);

    // Set P1.0 to output direction
    GPIO_setAsOutputPin(
        GPIO_PORT_P1,
        GPIO_PIN0
        );
```

```
CDT Build Console [blinkled_msp432p401r_MSP_EXP4
--display_error_number --gen_func_subsec
-m"blinkled_msp432p401r_MSP_EXP432P401R_
-i"C:/ti/simplelink_msp432p4_sdk_3_40_01
-i"C:/ti/ccs1000/ccs/tools/compiler/ti-c
--display_error_number --warn_sections
```

Then you should see this:


workspacev10 - blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs/main.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Variables
Expressions
Registers
main.c
Console

```
blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Suspended - HW Breakpoint)
    main() at main.c:55 0x0000019C
    _c_int00_noinit_noargs() at boot_cortex_m.c:133 0x000001EE  (_c_int00_noinit_noargs does not contain frame information)
```

Name	Type	Value	Location
i	unknown	Memory map prevented reading 0x20010000	

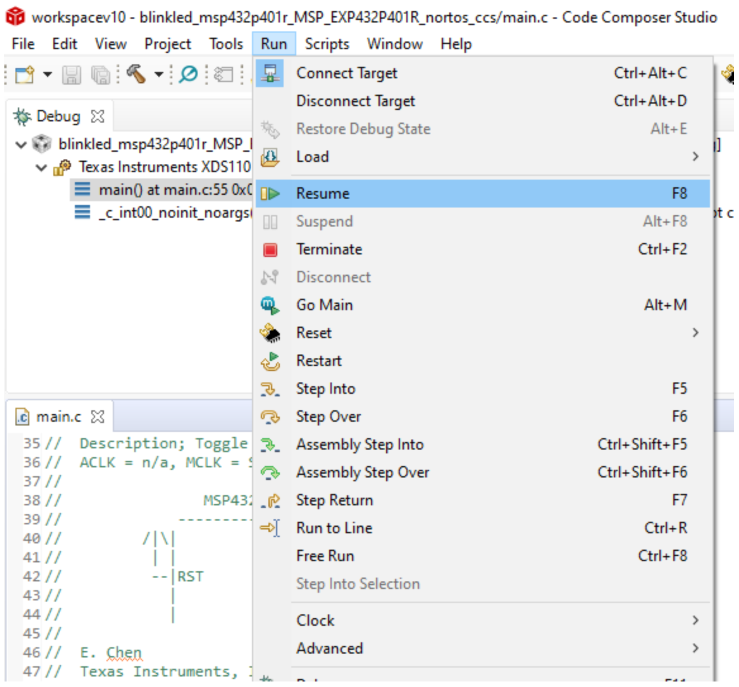
```
//  Description; Toggle P1.0 inside of a software loop.
//  ACLK = n/a, MCLK = SMCLK = default DCO
//
//                MSP432P4xx
//             -----------------
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|-
//            |                 |
//            |            P1.0|-->LED
//
//  E. Chen
//  Texas Instruments, Inc
//  March 2015
//  Built with Code Composer Studio v6
//******************************************************************************

#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

int main(void)
{
    volatile uint32_t i;

    // Stop watchdog timer
    WDT_A_hold(WDT_A_BASE);

    // Set P1.0 to output direction
    GPIO_setAsOutputPin(
        GPIO_PORT_P1,
```

```
blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs
CORTEX_M4_0: GEL Output: Memory Map Initialization Complete
```

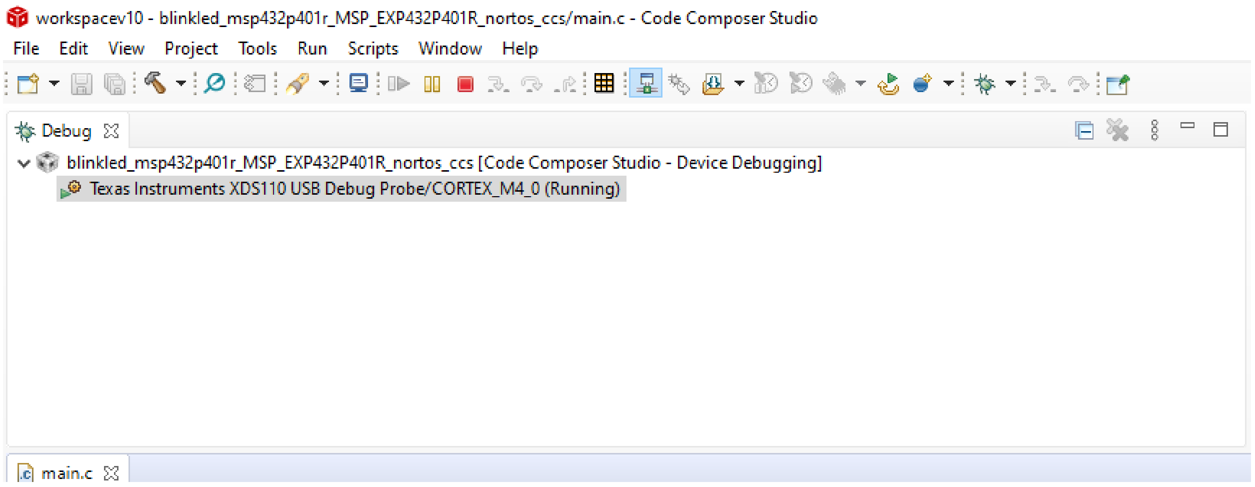
Notice that the entire view has changed. This is the debug Perspective. In the very top right hand of the IDE you can change between this and the CCS Edit perspective, which is where the IDE started.

Now let’s look at the information provided by this perspective. In the middle is the main.c file, but notice that the line right after the int main(void) line is highlighted. Also note the Console output at the bottom. It simply tells us that the connection has been made and that the file is downloaded. At the top of the screen is the Debug Output, this tells you that the code is running on the MSP432Rp401R, and is in the main function.

The code is not running. It turns out that the default mode downloads the code, starts running, and then stop right after the main(void) function is hit. To continue running Run->Resume (there are shortcuts for all of this you can learn.)


workspacev10 - blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs/main.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
blinkled_msp432p401r_MSP_
Texas Instruments XDS110
main() at main.c:55 0x0
_c_int00_noinit_noargs
main.c
Connect Target Ctrl+Alt+C
Disconnect Target Ctrl+Alt+D
Restore Debug State Alt+E
Load
Resume F8
Suspend Alt+F8
Terminate Ctrl+F2
Disconnect
Go Main Alt+M
Reset
Restart
Step Into F5
Step Over F6
Assembly Step Into Ctrl+Shift+F5
Assembly Step Over Ctrl+Shift+F6
Step Return F7
Run to Line Ctrl+R
Free Run Ctrl+F8
Step Into Selection
Clock
Advanced

Now the code should be running. The Debug window should now show:


workspacev10 - blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs/main.c - Code Composer Studio

blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs [Code Composer Studio - Device Debugging]

Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Running)

And an LED should be blinking on the MSP432P401R. Hopefully you are now at least familiar with the Code Composer IDE.