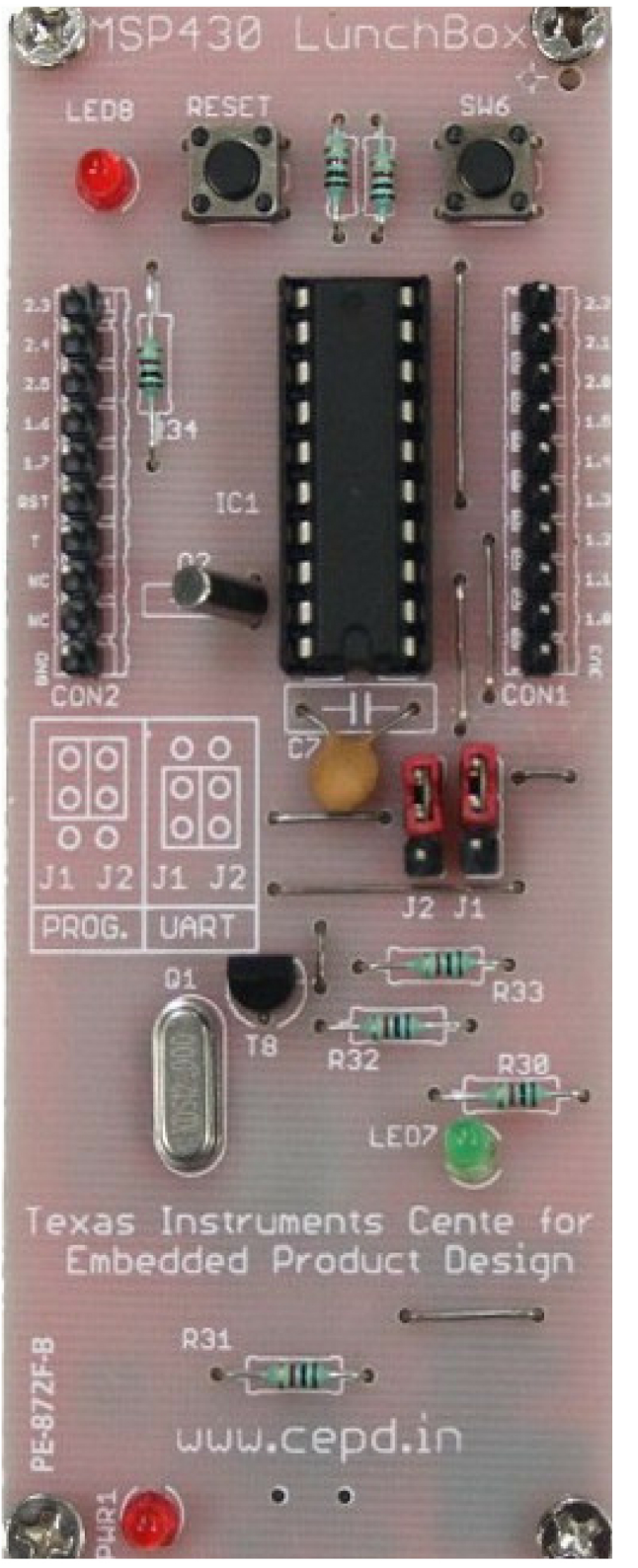
MSP430 LunchBox

A DIY PLATFORM FOR
LEARNING
MICRO-CONTROLLERS AND
PHYSICAL COMPUTING



cedtnsit.in

CONTENTS

Part I

MSP430 LUNCHBOX

1

MICROCONTROLLER KITS FOR THE MASSES

Arduino is one of the nicest things to happen to the DIY community in the last 10 years. It brought microcontroller usage within easy reach of non-specialists (from a technical viewpoint) such as artists, tinkerers, architects and musicians. Its low threshold of learning, simple and reliable programming environment brought it to the workbench of high school students as well. Apart from being a great, low learning-curve platform, what has helped Arduino to gain such traction worldwide, in no small measure, is the relative low cost too! Although the original Arduino boards cost tens of dollars, variants and knockoffs cost as little as $4! No wonder we find them popular with high and middle school students, everywhere.

A large majority of Arduino variants feature microcontrollers from the AVR family of Atmel (now acquired by Microchip). Some variants using the ARM Cortex M0 microcontrollers as well as higher end Intel processors are also available. Although TIs MSP430 is a strong competitor to AVR, boasting of superior features such as a 16-bit CPU, significantly lower power consumption and versatile peripherals,

it doesnt have a presence in the Arduino ecosystem. Even with significant software support in terms of the Energia (a fork of the Arduino IDE) which extends the simplicity of Arduino programming to TI microcontrollers, MSP430 has not been able to gain the same popularity as Arduino among enthusiasts. The relatively high cost of the MSP430 LaunchPad development kit ($10 and above) with no corresponding lower cost variants (as with Arduino family) is a likely deterrent.

What the MSP430 ecosystem needs to reach the masses is an extremely low cost entry-level platform which can stand up to the $3 Arduino variants. Such a platform is what we set about trying to design.

2

MICROCONTROLLER ESSENTIALS

Typically, a microcontroller system requires 4 support elements- power supply, clock, reset and code download ability. Let's take the example of the Arduino: It is powered by a 5V supply, either provided directly through a USB port, or from an external DC source via an onboard voltage regulator. It uses a crystal oscillator (8/16 MHz depending on the variant) for the system clock. It has a push button switch as well as a clever mechanism attached to the USB to Serial converter chip to reset the microcontroller. The user program is downloaded on to the chip from the Arduino IDE using the above mentioned USB to Serial Bridge, aided by a bootloader program which has to be manually loaded into each fresh chip.

3

INEXPENSIVE MSP430 EVALUATION KIT: THE MSP430 LUNCHBOX

With the MSP430, the job is a whole lot easier. Turns out, the MSP430 already has a built-in bootloader on-chip. All that is required is a mechanism to invoke the bootloader and send serial data to it- both of which can quite easily be achieved by a USB to UART Bridge. One of the cheapest USB to UART Bridge chips available in the market today is the CH340G - a full speed USB device that emulates a standard serial interface with speeds up to 2 Mbps and support for all modem handshaking signals- which costs less than half a dollar! CH340 is also one of the major reasons for the 4$ Arduinos. All this brings us to this - A $1 (conditions apply) MSP430 LunchBox- a low cost, maker-friendly microcontroller development platform featuring the 20-pin MSP430G2553 Value Line controller. The board supports any 14-pin or 20-pin DIP package MSP430 G series microcontroller, which a hobbyist can obtain for free through Texas Instruments' free sample programme. The entire bill of materials of the board, excluding the controller, is under $1. The PCB has been designed

to be a single sided, toner transfer friendly one, allowing enthusiasts to fabricate one for themselves at little or no cost. The photograph in below shows the early lab prototype of the MSP430 LunchBox.

4

COMPARING VARIANTS OF 'APPLES'

The MSP430 LunchBox has a feature set comparable to that of TI's own MSP430 LaunchPad Development Kit and can quite easily rival Arduino. Here's a side by side comparison of the $1 MSP430 LunchBox, TI's MSP430 LaunchPad and an Arduino Nano.

Feature	**$1 MSP430 LunchBox**	**MSP430 LaunchPad**	**Arduino Nano**
Microcontroller	MSP430G2553 & others	MSP430G2553 & others	ATMega328
CPU Architecture	16-bit	16-bit	8-bit
Operating Voltage	3.3V	3.3V	5V
Operating Clock	10 kHz to 16 MHz	10 kHz to 16 MHz	10 kHz to 16 MHz
Operating Current	4.5 mA @ 16 MHz	4.5 mA @ 16 MHz	15 mA @ 16 MHz
Programming	Factory UART BSL	Onboard Spy-Bi-Wire	Custom Bootloader
Debugging	Not supported	Spy-Bi-Wire debugger	Not supported
Supported IDEs	CCS, Energia	CCS, Energia	Arduino, Atmel Studio
Available I/Os1	14	16	20
Analog Inputs	8	8	6
PWM Outputs	6	6	6
Peripherals	1 LED, 1 Switch, UART	2 LEDs, 1 Switch, UART	1 LED, UART
Cost	**$1**	**$10**	**$4**

5

LUNCHBOX (HARDWARE)

The LunchBox functionality as listed above, can be seen in a block diagram format here:

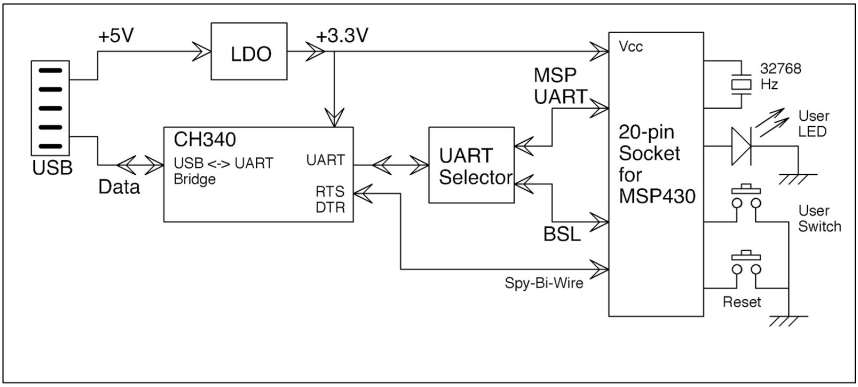
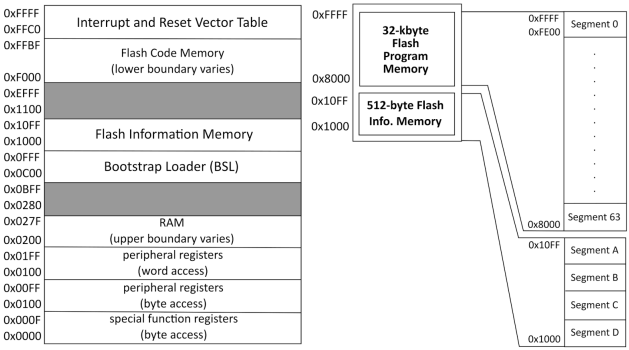


Figure 1.: Block diagram of MSP430 LunchBox

Challenges Faced

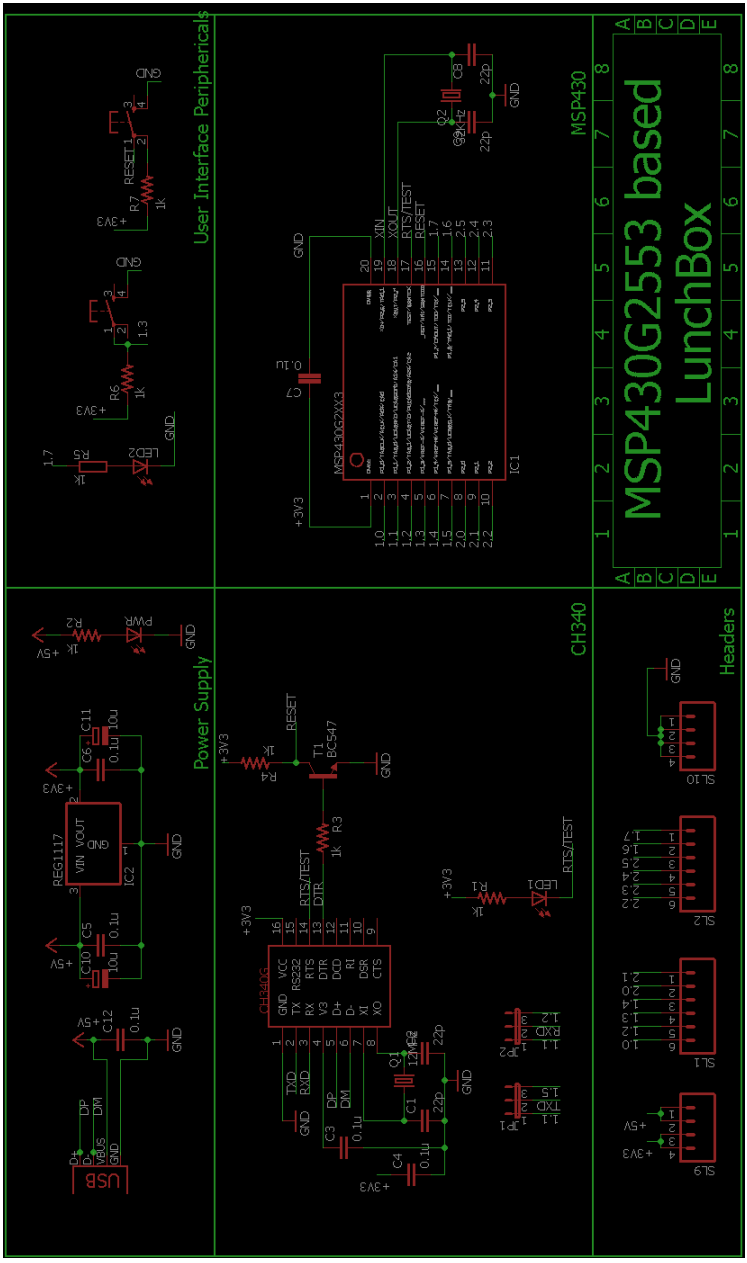
1. While the MSP430 has an in-built UART bootstrap loader (BSL), they are not brought out on the same pins as the standard UART interface of the MSP430. This meant that a provision for switching the CH340 USB to UART Bridge between the BSL UART and the MSP UART peripheral - implemented using a pair of shorting jumpers on board - had to be made.

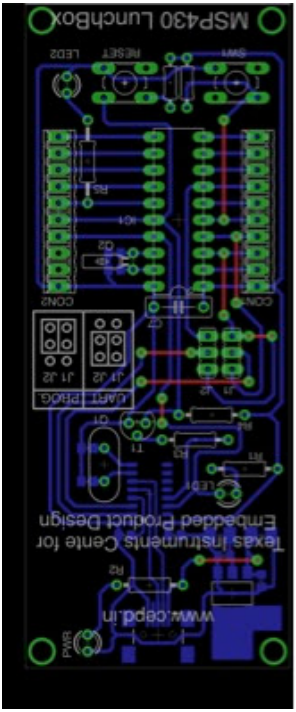
2. Another major challenge was on the software side - the UART BSL utility provided by Texas Instruments is quite outdated and contains a few bugs. One of the most critical is the issue with how the BSL Utility handles the flash memory. The flash memory on MSP430 is divided into two parts - the code memory, which contains the user code and the information memory, which typically contains data such as calibration constants, Digitally Controlled Oscillator (DCO) settings etc.

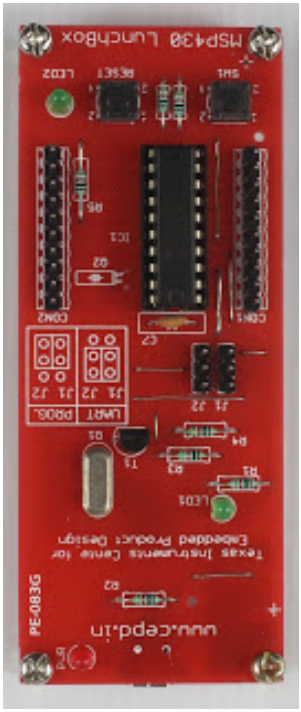


> The calibrated DCO settings, which are required to generate accurate high speed clocks in the MSP are stored in Segment A of the Flash Information Memory (INFO-A). Even though the BSL Software Utility provides an option to preserve the contents of the INFO-A segment, it does not seem to work properly and ends up erasing the entire information and code memory when invoked. Without the DCO constants, it is extremely difficult to implement UART communication due to errors in the system clock. As the bug fixes on an outdated software utility was quite a cumbersome task, we decided to implement a hardware solution to this issue - an external 32.768 kHz crystal oscillator which can be used as an accurate clock source for implementing UART communication. This ends up reducing the number of I/O pins available to the user by 2 pins, but also gives the advantage of having a crystal oscillator that can be easily used for real-time clock applications.

Here is the complete schematic diagram of the LunchBox. It shows the power supply for the MSP430 3.3V, derived from the 5V available on the USB connector and also available on the output headers. The clock for the microcontroller generated using an external crystal of 32.768 KHz frequency, connected to the Xin and Xout pins of the MSP430. The user interface peripherals are a simple pushbutton and an LED apart from the Reset switch. Jumpers J1 and J2 allow the user to switch the CH340 USB to UART bridge from the MSP's UART peripheral pins to MSP's BSL pins. These jumpers are manipulated manually.


MSP430G2553 based
LunchBox
User Interface Peripherals
Power Supply
CH340
MSP430
Headers
MSP430G2XX3
IC1
REG1117
IC2
CH340G
RESET
RTS/TEST
LED1
LED2
PWR
USB
SL1
SL2
SL9
SL10
+3V3
+5V
GND





Here are the PCB layout and a photo of the actual board.

6

WRITING YOUR FIRST CODE FOR LUNCHBOX

Writing code for LunchBox is similar to writing code for any other TI-LaunchPad using CCS. The only thing which you need to do is create a template project first (see Appendix B), copy it and then remane it as per your liking.

We are assuming here that you have the template project ready with you and only going to add code to your .c file in your project.

When you open your .c file, you will have following code already written in it.

```
#include <msp430.h>

/**
 * main.c
 */
int main(void)
{
  WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

  return 0;
}
```

Blink an LED

In this example, we will demonstrate how to upload the code by blinking the On-Board LED connected to pin 1.7 of the microcontroller. Write or Copy the code below in your .c file.

```
#include <msp430.h>


/**
 * blink.c
 */
void main(void)
{
  WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer
  P1DIR |= 0xFF;              // configure P1.7 as
    output

  volatile unsigned int i;    // volatile to prevent
    optimization

  while(1)
  {
    P1OUT ^= 0xFF;             // toggle P1.7
    for(i=10000; i>0; i--);    // delay
  }
}
```

Next, right click on the project in the project explorer, locate "Rebuild project" option and click on it. It will rebuild your project and upload it to your connected LunchBox. As soon as the blinking of the Green LED stops (indicating that the code is being uploaded), your code has been uploaded to the microcontroller, and the red LED will start blinking.

Part II

APPENDIX

A

GETTING STARTED WITH CODE COMPOSER STUDIO (CCS)

Introduction

Welcome to Code Composer Studio v6! Code Composer Studio v6 is a major new release of Code Composer Studio (CCS) that is based on the Eclipse open source software framework. The Eclipse software framework is used for many different applications but it was originally developed as a open framework for creating development tools. We have chosen to base CCS on Eclipse as it offers an excellent software framework for building software development environments and is becoming a standard framework used by many embedded software vendors. CCSv5 combines the advantages of the Eclipse software framework with advanced embedded debug capabilities from Texas Instruments resulting in a compelling feature rich development environment for embedded developers.

Obtaining CCS

1. Right-click on the link below and open another window. Choose the desired Code Composer Studio v6 release:
   `http://processors.wiki.ti.com/index.php/Download_CCS`

2. Clicking on the link of the desired version opens the my.ti.com registration page. Fill in the correct login information and click Log In.

3. The next page is the Export approval, which is required for this product. Fill in with the required information and hit Submit.

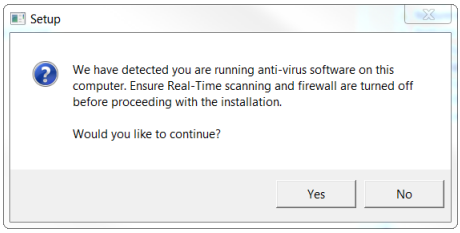
4. A message with the subject TI SOFTWARE DOWNLOAD: APPROVED and the name of the installer file will be sent to the provided e-mail address. This message contains a limited-time link to download the software.

5. Click on the link and a browser window is opened to perform the download.

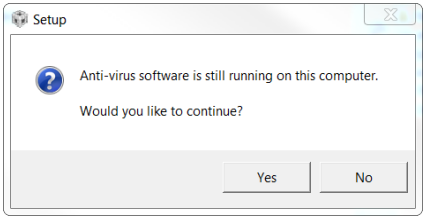
6. After the download is complete, either run the small installer or unzip the full installer file to a temporary directory and start running.

Installing CCS

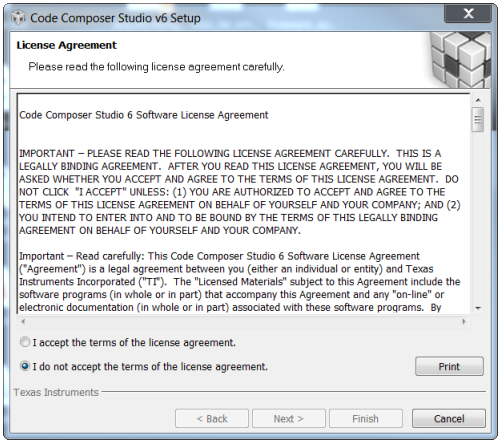
1. In Windows, double-click on it and click on the button Extract All Files. Write down which directory the installer was extracted.
2. In Windows, double click on the installer <ccs setup 6.x.x.xxxxx.exe> (the x represents the software version and may vary).
3. The installer will open. If you have antivirus software enabled (which is very common to cause problems during install), the following pop-up will appear:



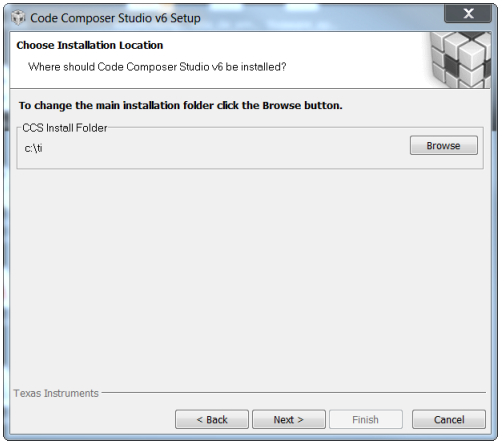
If you do not disable it, the second pop-up will be shown (yes, it is very important to disable it):



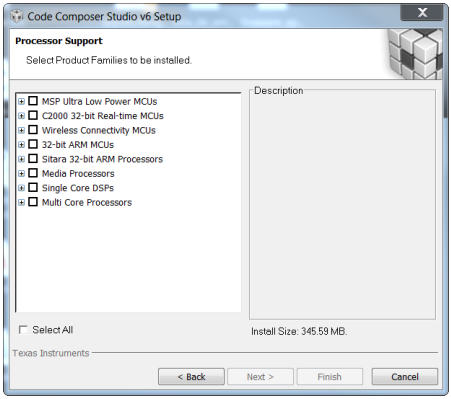
4. The next screen is the license agreement. It has to be accepted before moving to the next step.



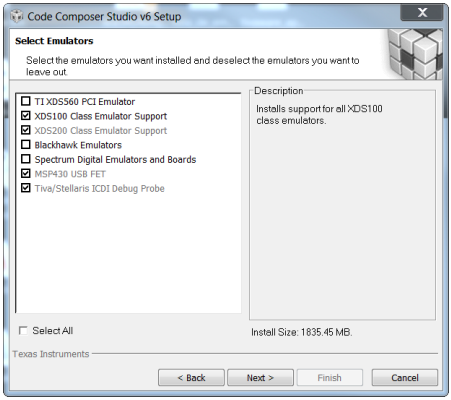
5. Next step is to select the desired installation location



6. Next step is to select the desired device families and its components



7. The next screen allows selecting the emulator support to be installed. If needed, additional emulator support can be installed later.



8. Clicking next shows some of the applications and add-ons available at the CCS App Center. Any selection will cause CCSv6 to prompt you to install them the first time it runs (additional details on the next section).

9. Click Finish to start the install process.

B

PREPARING YOUR CODE TEMPLATE

Programming LunchBox requires changing the default build options of Code Composer Studio (CCS). And doing the same changes every time you start working on a project will take up unnecessary time. For that purpose, its best to have a template ready in your project explorer. Steps for preparing your template are given below:

1. Start by opening the following link and downloading the zip file:
   `http://www.ti.com/lit/zip/slaa535.`

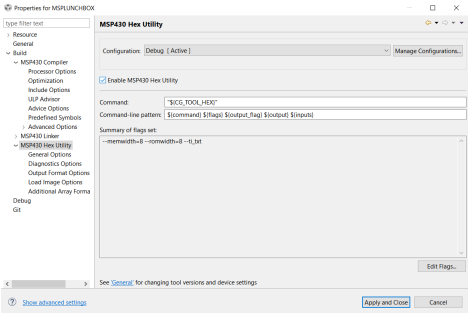
   Extract the zip file and save it in a folder which you can remember.

2. In the extracted folder, go to TestScripts -> Exe. Locate BSLDEMO2.exe and copy the location of that folder.

3. Next open CCS and create an empty project for MSP430G2553.

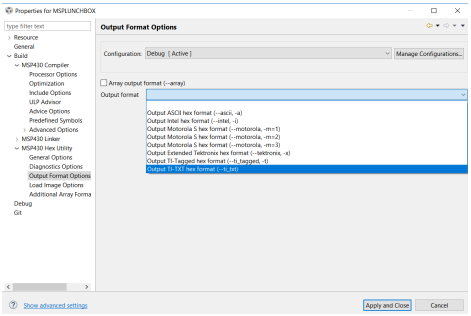
4. Right click on the project and open properties. A new tab will be opened.

5. In the new tab, locate "MSP430 Hex Utility" and click on it.



6. Now select the Output Format Options part and choose

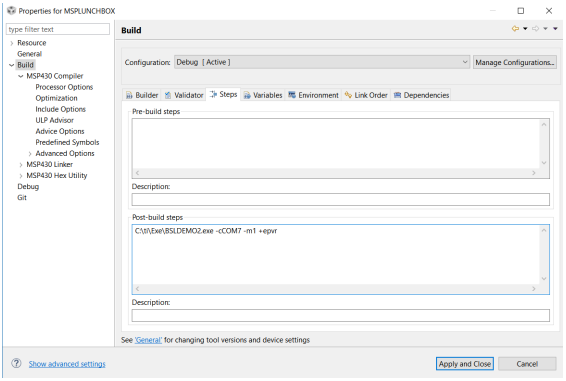
   Output TI-TXT hex format



7. Go to build option in properties window.Paste the location of BSLDEMO2.exe, add a space type the following in front of it:

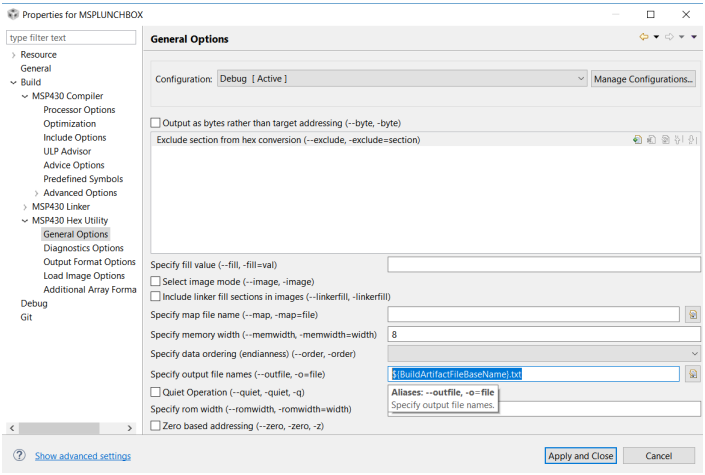
-cCOM6 -m1 -epvr

(Note that value of Com-port may be different for your system. Please check yours in the device manager.)



8. Go to **General options** under **MSP430 Hex Utility** and copy the text given under the dialogue box **Specify output file names**

$BuildArtifactFileBaseName.txt



9. Finally the Post-bulid steps under **Build** settings should include the text as given

10. Click "Apply and close" in properties tab.

Your template is ready to be used for further projects.

C

EPILOGUE

How does one justify a BOM cost of 1$ for the LunchBox? The trick lies in ordering free samples from TI: the relevant MSP430 G series microcontroller and the LM1117 linear regulator. The only component you may need to purchase would be the CH340 USB to UART bridge and currently, this sells for 50 cents in volumes. The rest of the components are easily available in your, the electronics enthusiast's components box!