Introduction to Embedded System Design

Getting acquainted with GIT, CCS Installation and Embedded C

Dhananjay V. Gadre
Associate Professor
ECE Division
Netaji Subhas University of Technology, New Delhi

Badri Subudhi
Assistant Professor
Electrical Engineering Department
Indian Institute of Technology, Jammu

Version Control System: GIT

Software link

- https://git-scm.com/downloads
- Follow normal installing steps

Initial configuration of user

- git config --global user.name "Username"
- git config --global user.email "email.id@domain.com"

Create Repositories

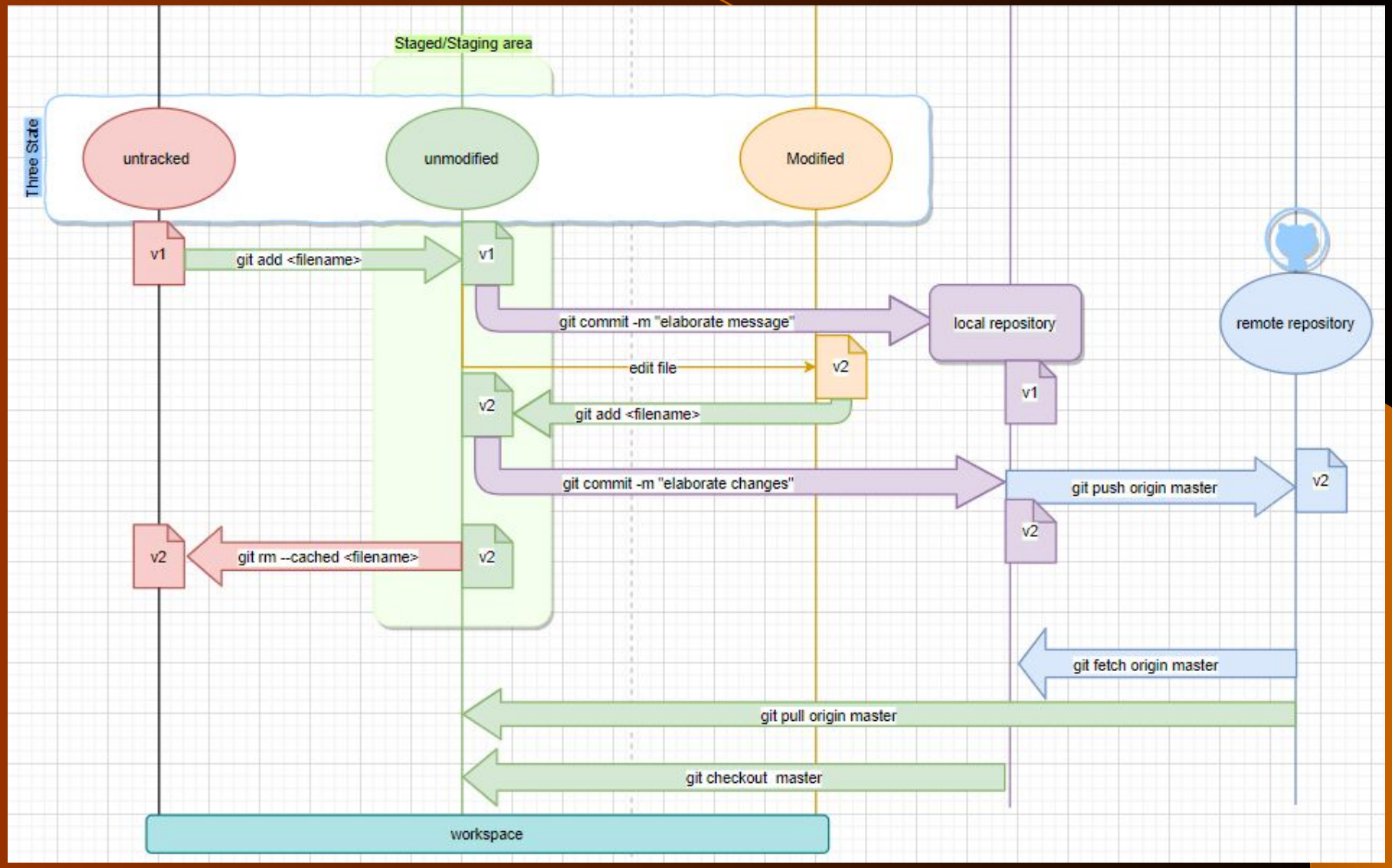
- git init

  or
- git clone <url>
  - For example: git clone https://github.com/ticepd/EmbSysDesign_NPTEL_Course.git

Handling changes

- git add <file>
- git commit -m “elaborate changes message”
- git log
- git log --follow <file>

Basic workflow of GIT



Synchronize changes

- git fetch
  - Downloads all history from the remote tracking branches.
- git pull
  - Updates your current local working branch with all new commits from the corresponding remote branch on GitHub.

Clone remote repository for this series

git clone
https://github.com/ticepd/EmbSysDesign_NPTEL_Course.git

Download files directly

https://github.com/ticepd/EmbSysDesign_NPTEL_Course/archive/master.zip

Setting up CCS

OS:- Windows

Setting up CCS : Download page

Code Composer Studio Downloa

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

Code Composer Studio Version 9 Downloads

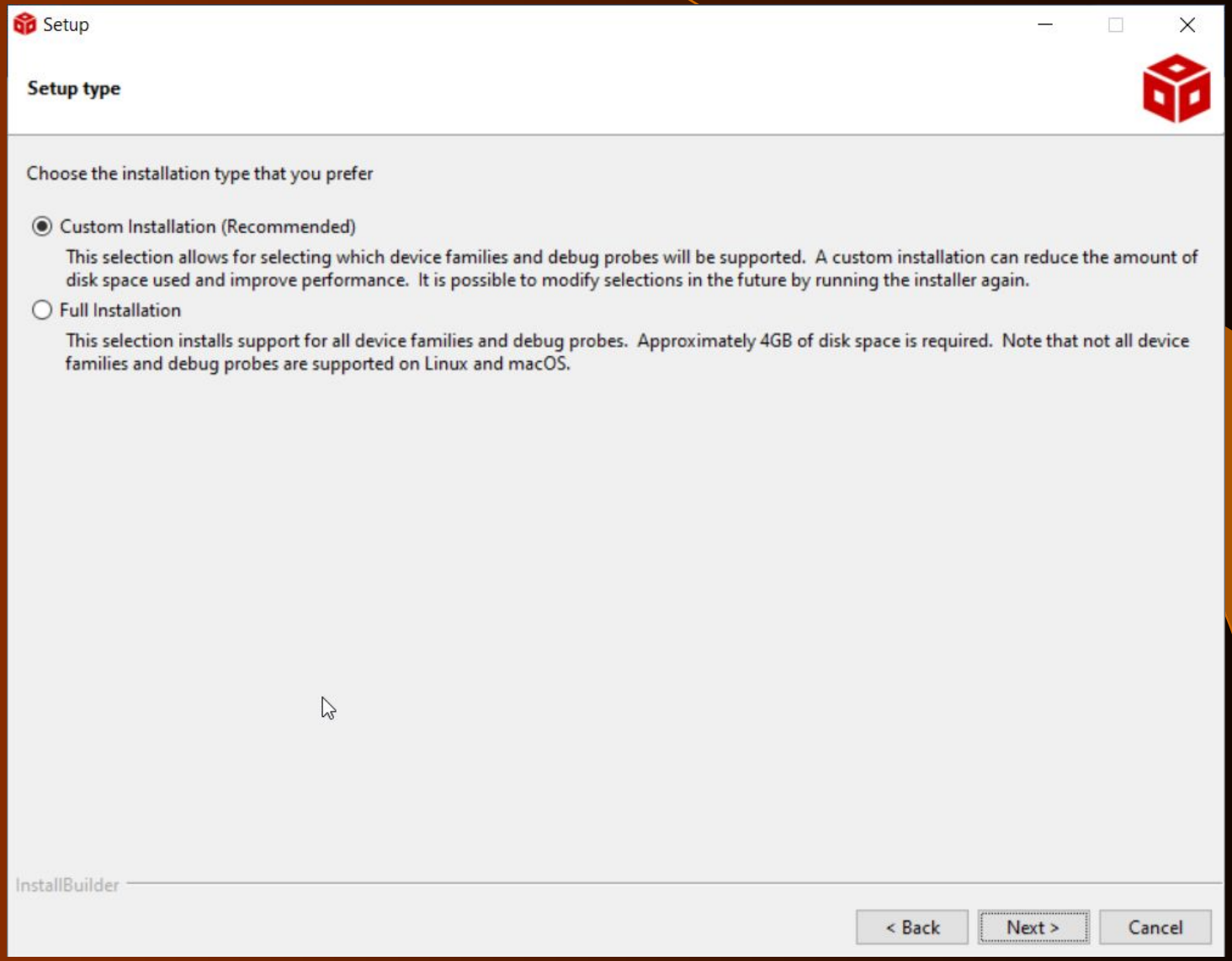
There are two types of installers:

- 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 800MB) so you may then uncompress it then select what you require to install. Formerly known as offline installers.

Release	Build #	Date	Download	Notes
9.3.0	9.3.0.00012	Dec 19, 2019	**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	• **New/Notable In This Release (9.3.0.00012):** • Release notes • Windows support: Code Composer Studio is now supported only on 64bit Windows machines. • Mac OS installers are now distributed as signed and notarized disk image (DMG) files. • Bugfixes for IDE, DVT, Debugger • MCU Compilers LTS 18.12.4 • C6000 Compiler 8.3.5 • MSP GCC v8.3.1.25 • XDCTools 3.60.02 • Device support updates • TI EMU19_M11 v8.4.0.00006

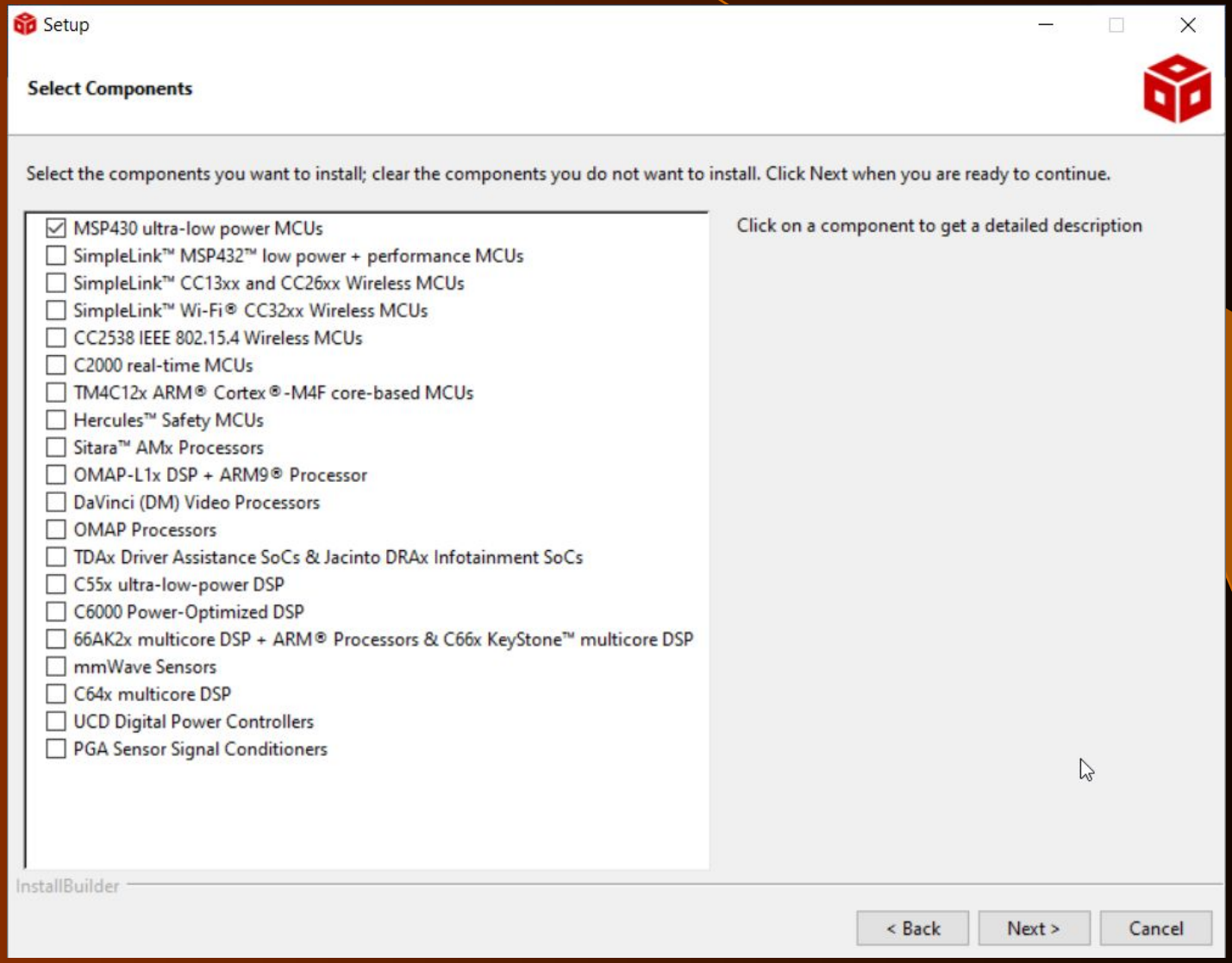
OS:- windows

Setting up CCS



OS:- windows

Setting up CCS : choose MSP430



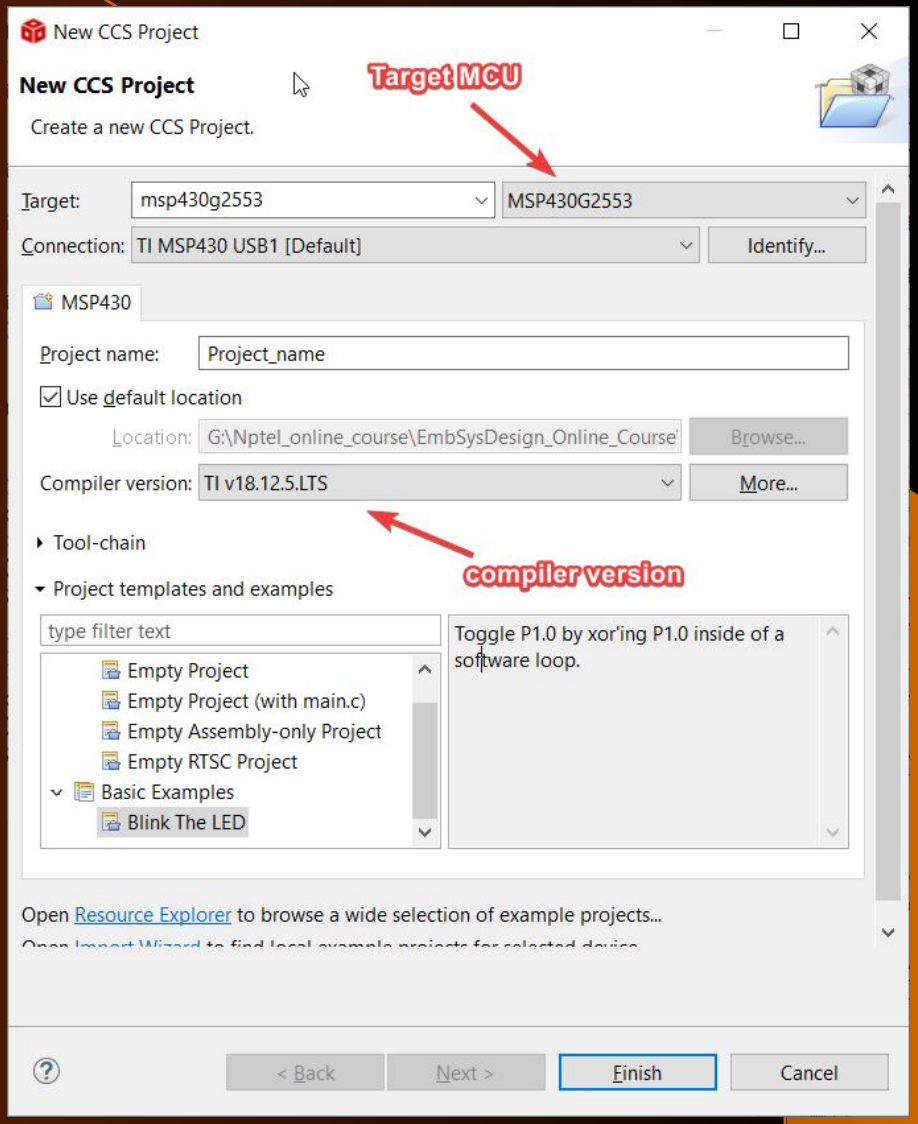
Import project in CCS after cloning from Github

- On starting CCS for the first time, Initial projects (provided on Github and as shown in upcoming videos) will not be shown inside CCS project explorer.
- To import those existing projects from git cloned path to CCS, Proceed with following steps inside CCS,
    - Go to File > Import.
    - General > Existing projects into workspace
    - Select root directory to Software\Examples_Msp430G2553_LunchBox in cloned directory
    - Refresh > Select All
    - Finish

Setting up a New Project for MSP430G2553

Navigate using following steps:-

- File > New > CCS project



Automatically generated settings with project

Properties for Exp02_LunchBox_HelloBlink

type filter text

- Resource
- General
- Build
  - MSP430 Compiler
    - Processor Options
    - Optimization
    - Include Options
    - ULP Advisor
    - Advice Options
    - Predefined Symbols
    - Advanced Options
  - MSP430 Linker
  - MSP430 Hex Utility
- Debug
- Git
- Project Natures

Predefined Symbols

Configuration: Debug Manage Configurations...

Pre-define NAME (--define, -D)

__MSP430G2553__

used during conditional compilation

LLVM Optimization Level (--llopt)

Undefine NAME (--undefine, -U)

_INLINE

location

Quick Access

msp430.h msp430g2553.h

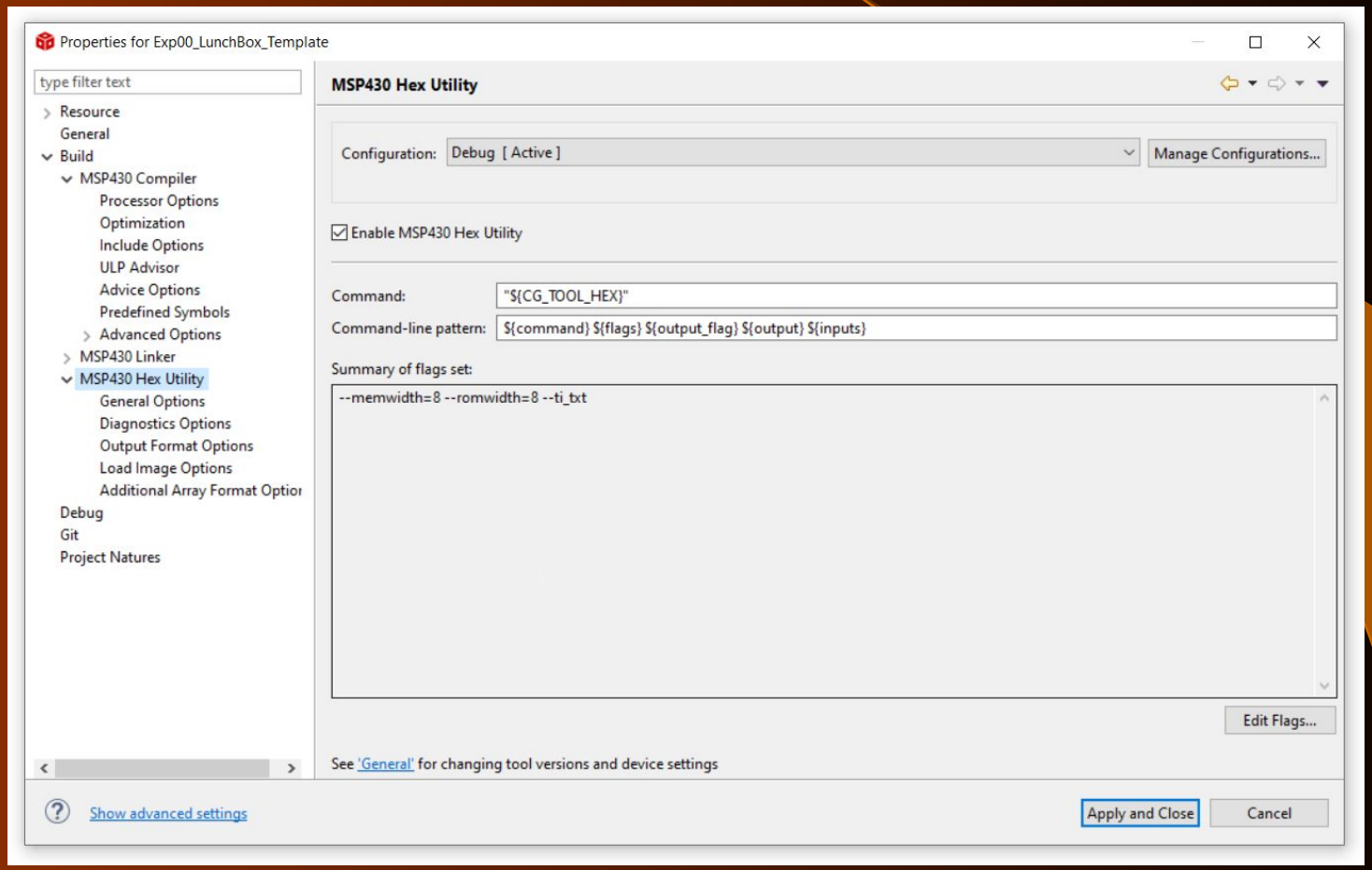
Target MSP430G2553

```
1418
1419 #elif defined (__MSP430G2453__)
1420 #include "msp430g2453.h"
1421
1422 #elif defined (__MSP430G2553__)
1423 #include "msp430g2553.h"
1424
1425 #elif defined (__MSP430G2203__)
1426 #include "msp430g2203.h"
```

OS:- windows

BSL settings in CCS

Step 1

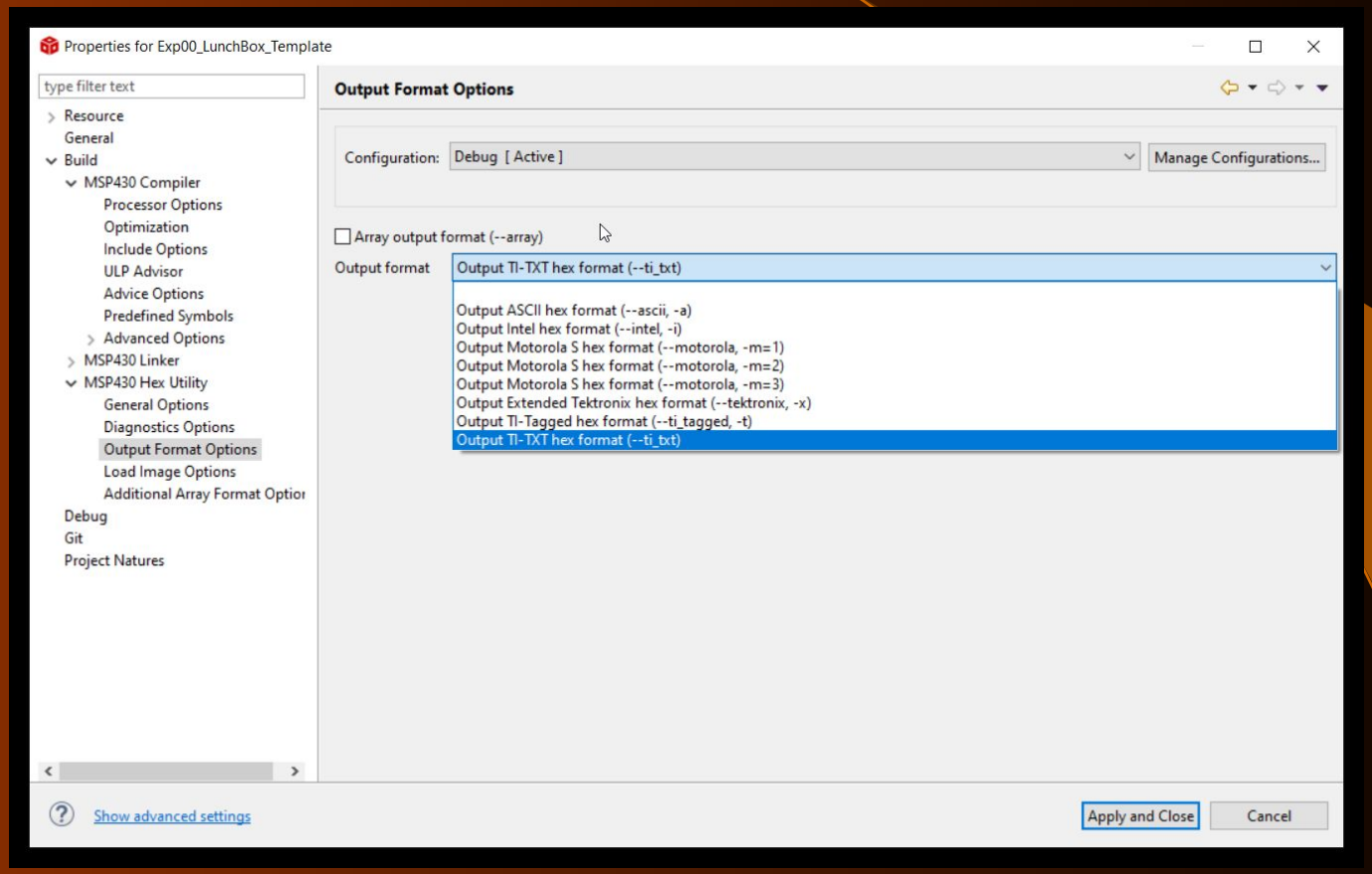


- Project Properties > Build > MSP430 Hex Utility > Enable MSP430 Hex Utility

OS:- windows

BSL settings in CCS

Step 2



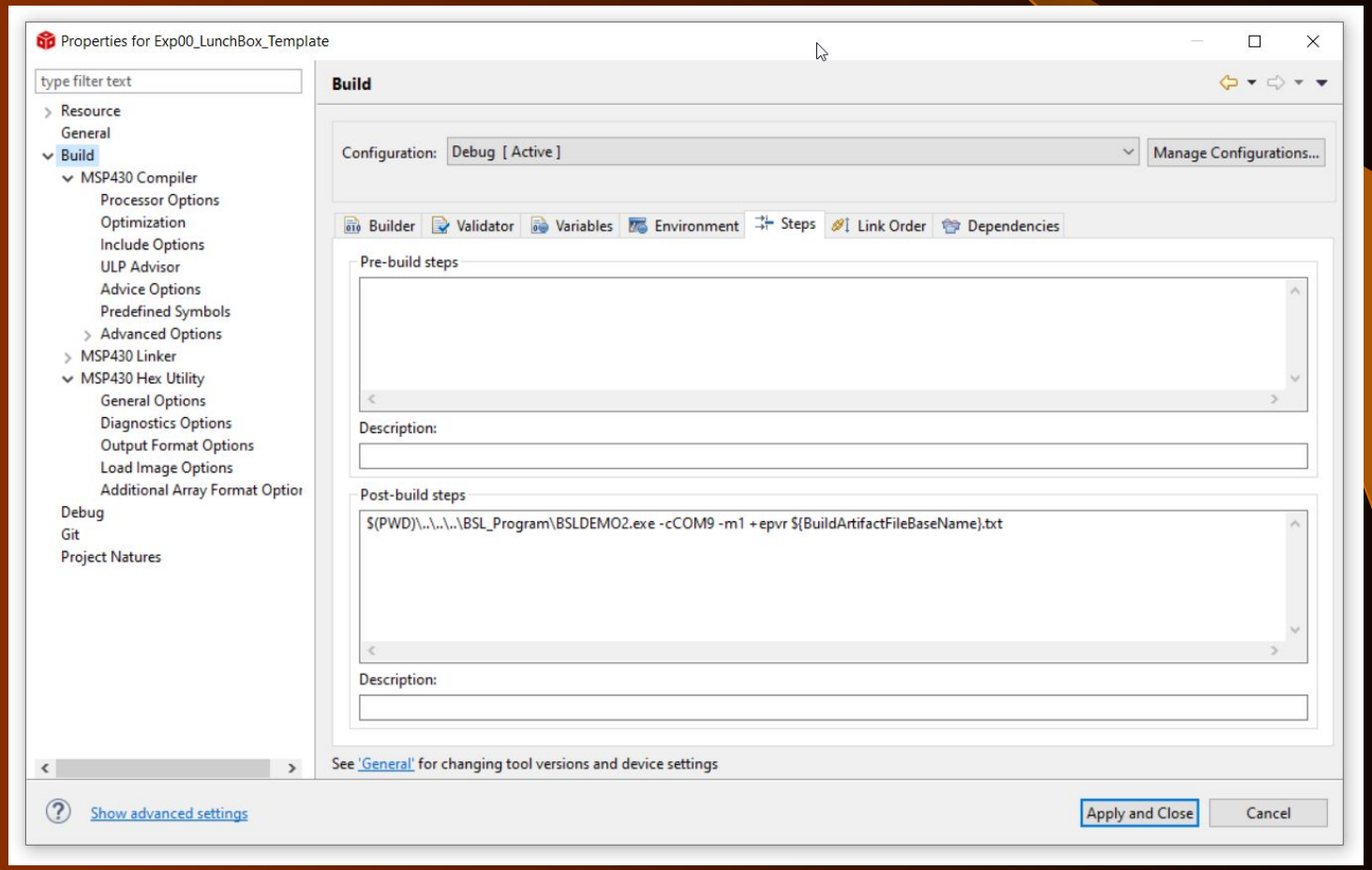
- Project Properties > Build > MSP430 Hex Utility > Output Format Options > Output TI-TXT hex format

Step 3

OS:- windows

BSL settings in CCS

- Project Properties > Build > Steps tab > Post-build steps
- $(PWD)\..\..\..\BSL_Program\BSLDEMO2.exe -cCOM9 -m1 +epvr ${BuildArtifactFileBaseName}.txt



MSP430 C/C++ compiler

The TI compiler accepts C and C++ code conforming to the International Organization for Standardization (ISO) standards for these languages. The compiler supports both the **1989 and 1999 versions of the C language** and the 2014 version of the C++ language.

Level :- For beginner

Keywords in C

The MSP430 C/C++ compiler supports all of the standard C89 keywords, including const, volatile, and register. It also supports all of the standard C99 keywords, including inline and restrict. It also supports TI extension keywords __interrupt, and __asm. Some keywords are not available in strict ANSI mode.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Main function

- Entry point
- Its compulsory.
- Good practice is not to exit it. (use while loop)

General main() code flow

- Configure WDT
- Set up the clock
- Configure ports and peripherals
- Enable Interrupts

Structure of Embedded C code

Exp00_LunchB... | Exp02_LunchB... | Exp40_LunchB... | Exp32_LunchB...

```c
#include <Library/LunchboxCommon.h>
#include <msp430.h>
#include <stdio.h>
#include <string.h>
#include <inttypes.h>
/*
 * An example of multi-line comments
 * @brief entry point for the code
 * */
int main(void)
{ //! Stop Watchdog (Not recommended for code in production and devices working in field)
    WDTCTL = WDTPW | WDTHOLD;

    initialise_SerialPrint_on_lunchbox(); // a function
    int x = 0;
    while (1)
    {
        x++;
        printf("Hello world %d \r\n", x);

        int i;
        for (i = 0; i < 20000; i++);

    }
}
```

Elements of Embedded Programing

Structure of the program:

A program generally has the following parts

- Library
- Main function
- Subroutines
- Interrupt Service Routines

C keywords provided to manipulate memory allocations

- Variable types (eg. int, char, uint8_t, bool,enum)
- Type modifiers
  - size manipulation (eg. short, long)
  - sign manipulation (eg. unsigned, signed)

C keywords provided to manipulate memory allocations

- Storage classes
  - Manipulate lifetime and scope
  - Examples
    - auto
      - It allocates local variable in stack memory
    - static
      - Data will persist till program execution
    - extern
      - Allow access outside current scope/file
    - register
      - Allocates memory in cpu register. It's not so common, compiler does it better

Variables qualifier

- const : Declaring a variable as a constant means that it's value cannot be modified by the program.
- volatile: By declaring a variable as volatile, the compiler gets to know that the value of the variable is susceptible to frequent changes (with no direct action of the program) and hence the compiler does not keep a copy of the variable in a register (like cache). This prevents the program to misinterpret the value of this variable.
- As a thumb rule: variables associated with input ports should be declared as volatile.

Data types in C

Type	Size	Alignment	Representation	Range	
				Minimum	Maximum
signed char	8 bits	8	Binary	-128	127
char	8 bits	8	ASCII	0 or -128 (1)	255 or 127 (1)
unsigned char	8 bits	8	Binary	0	255
bool (C99)	8 bits	8	Binary	0 (false)	1 (true)
_Bool (C99)	8 bits	8	Binary	0 (false)	1 (true)
bool (C++)	8 bits	8	Binary	0 (false)	1 (true)
short, signed short	16 bits	16	2s complement	-32 768	32 767
unsigned short	16 bits	16	Binary	0	65 535
int, signed int	16 bits	16	2s complement	-32 768	32 767
unsigned int	16 bits	16	Binary	0	65 535
long, signed long	32 bits	16	2s complement	-2 147 483 648	2 147 483 647
unsigned long	32 bits	16	Binary	0	4 294 967 295
long long, signed long long	64 bits	16	2s complement	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
unsigned long long	64 bits	16	Binary	0	18 446 744 073 709 551 615
enum	varies (2)	16	2s complement	varies	varies
float	32 bits	16	IEEE 32-bit	1.175 494e-38 (3)	3.40 282 346e+38
double	64 bits	16	IEEE 64-bit	2.22 507 385e-308 (3)	1.79 769 313e+308
long double	64 bits	16	IEEE 64-bit	2.22 507 385e-308 (3)	1.79 769 313e+308
function and data pointers	varies (see	16			

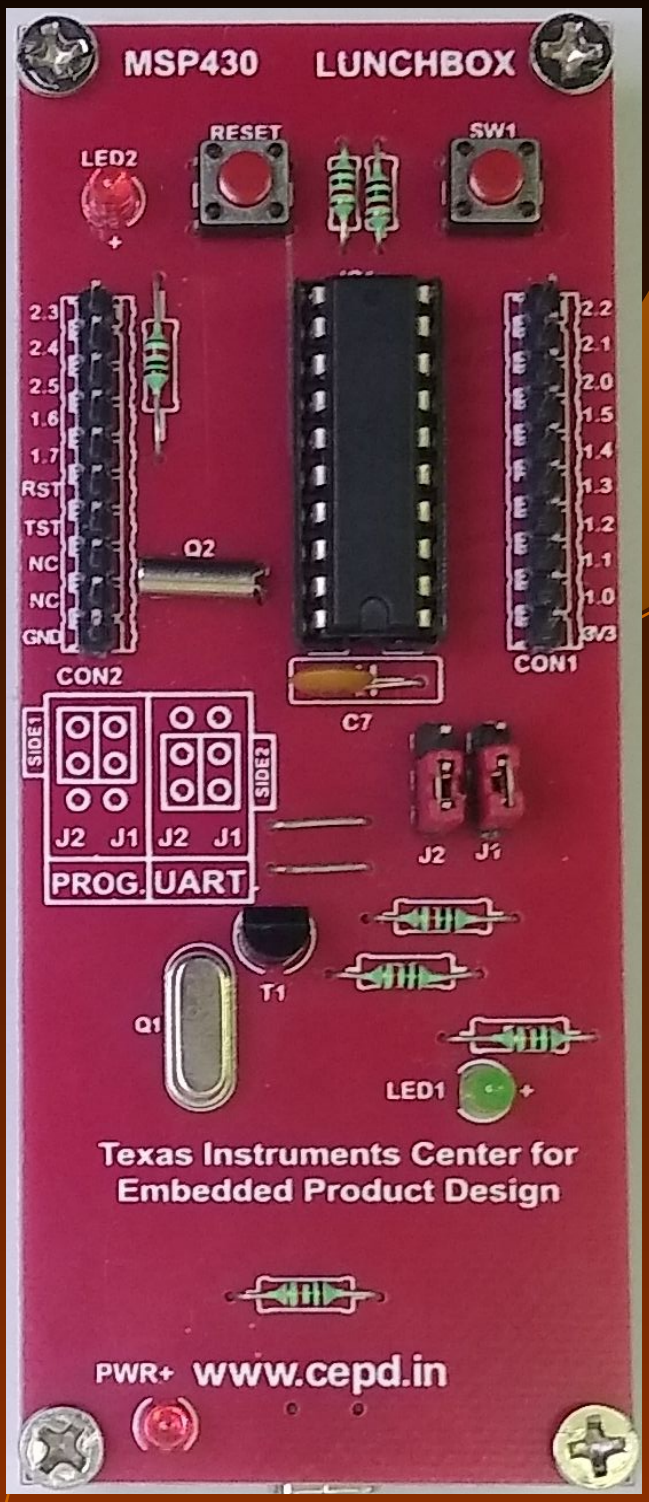
Source :- MSP430 Optimizing C/C++ Compiler v18.1.0.LTS User's Guide (Rev. R)

uintXX_t Data types in C

These data types (Example → uint8_t, uint16_t) are included in the header file - “inttypes.h”.

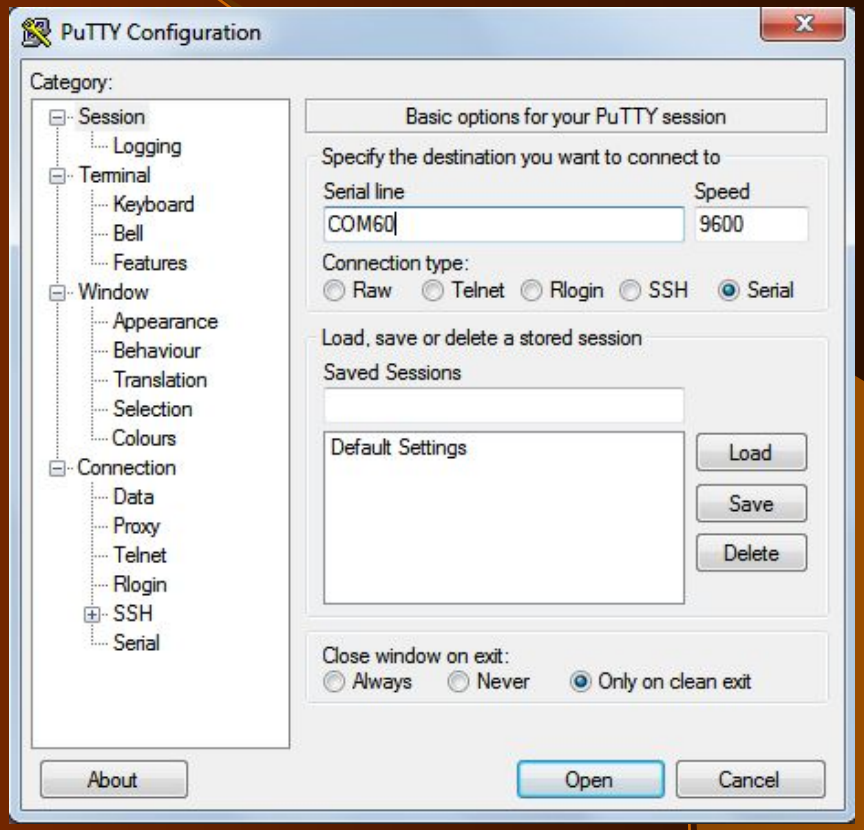
It's very useful when you do computations on bits, or on specific range of values (in cryptography, or image processing for example) because you don't have to "detect" the size of a long, or an unsigned int, you just "use" what you need. If you want 8 unsigned bits, use uint8_t like you did.

These are cross-platform. You can compile your program on a 32-bits or a 16-bits controller, and you won't have to change the types.


MSP430 LUNCHBOX
RESET
SW1
LED2
2.3
2.4
2.5
1.6
1.7
RST
TST
NC
NC
GND
Q2
2.2
2.1
2.0
1.5
1.4
1.3
1.2
1.1
1.0
3V3
CON2
CON1
C7
SIDE1
SIDE2
J2 J1
J2 J1
PROG.
UART
J2 J1
T1
Q1
LED1
Texas Instruments Center for
Embedded Product Design
PWR+ www.cepd.in

Download and Install PuTTY

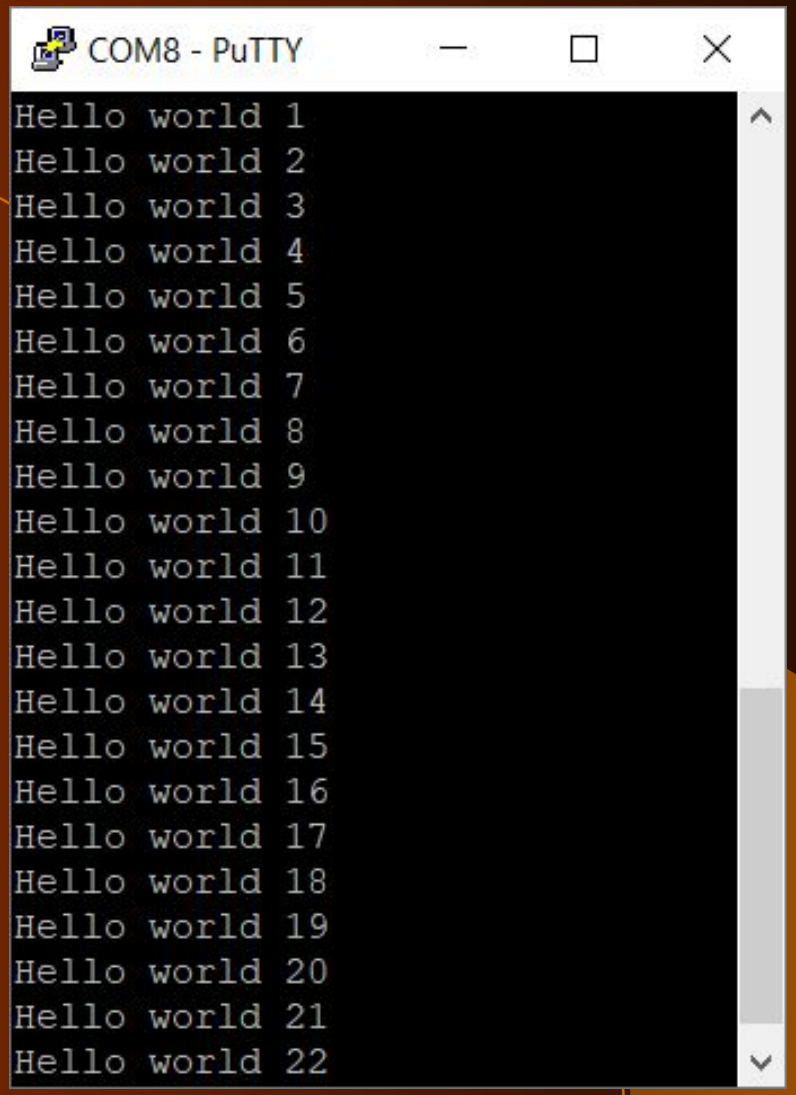
PuTTY is a free and open-source terminal emulator, serial console and network file transfer application.
PuTTY can be used as a serial terminal for this Code Example.



Download link :-
https://www.putty.org/

Steps to invoke :-
double click, change port number and click open

Debugging using Serial Monitor



Bit level access tricks on registers using C

- <u>Set Bit:</u>
  P1DIR = 8; //0000 1000
  P1DIR |= (1 << 2); //0000 0100
  P1DIR |= (BIT7 + BIT6) // 1100 0000
  → Result 1100 1100
- <u>Clear Bit:</u>
  P1DIR = 204; //1100 1100
  P1DIR &= ~(1 << 3);
  → Result 1100 0100
- <u>Flip Bit:</u>
  P1DIR = 8; //0000 1000
  P1DIR ^= (1 << 5);
  → Result 0010 1000

- <u>Get Bit:</u>

```
P1DIR = 8;      //0000 1000
if ( (P1DIR  & (1 << 3)) > 0)
{
    // do this if the 3rd bit is set(1)
}
else
{
    // do this if the 3rd bit is not set(0)
}
```

Example of Embedded C: demo of Bit Manipulation

COM8 - PuTTY

```
x value          000000010
y value          000000100
z value          000001000



Bitwise operators in C

value of a & b          000000100
value of a | b          001100100
value of a<<1   010001000
value of a^b    001100000
value of (1<<3)         000001000
value of a|(1<<4)       001010100
```

Exp41_LunchBox_Introduction_to_Embedded_C.c

```c
#include <msp430.h>
#include <stdio.h>
#include <string.h>
#include <inttypes.h>
#include <Library/LunchboxCommon.h>
int main(void)
{   //! Stop Watchdog (Not recommended for code in production
    //and devices working in field)
    WDTCTL = WDTPW | WDTHOLD;

    initialise_SerialPrint_on_lunchbox(); // a function
    while (1)
    {
        uint8_t x = 2;
        uint8_t y = 4;
        uint8_t z = 8;
        printf("\r\n x value \t");
        decToBinary(x);
        printf("\r\n y value \t");
        decToBinary(y);
        printf("\r\n z value \t");
        decToBinary(z);
        printf("\r\n\r\n\r\n");
        printf("\r\n Bitwise operators in C \r\n");
        uint8_t a = 0b01000100; //68 or 0x44
        uint8_t b = 0b00100100; //36 or 0x24
        printf("\r\n value of a & b \t");
        decToBinary(a & b);
        printf("\r\n value of a | b \t");
        decToBinary(a | b);
        printf("\r\n value of a<<1 \t");
        decToBinary(a << 1);
        printf("\r\n value of a^b \t");
        decToBinary(a ^ b);
        printf("\r\n value of (1<<3) \t");
        decToBinary(1 << 3);
        printf("\r\n value of a|(1<<4) \t");
        decToBinary(a | (1 << 4));
        printf("\r\n\r\n\r\n");
        unsigned long delay;
        for (delay = 0; delay < 60000; delay++);
    }
}
```

Blink led example using msp430.h

Exp02_LunchBox_HelloBlink_with_msp430_h.c

```c
#include <msp430.h>
/*@brief entry point for the code*/
void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    //(*(volatile unsigned int *) 0x0120) = 0x5A00 | 0x0080;
    /* ! Stop Watchdog (Not recommended for code in production
     and devices working in field)
     */

    P1DIR |= BIT7;
//    (*(volatile unsigned char *) 0x0022) |= 0x80;
    // P1.7 (Red LED)

    volatile unsigned long i;
    while (1)
    {

        //P1OUT |= BIT7;
        P1OUT |= 0x80;                //Red LED -> ON
//        (*(volatile unsigned char *) 0x0021) |= 0x80;
        for (i = 0; i < 10000; i++)
            ;   //delay

        //P1OUT &=~ BIT7;
        P1OUT &= ~0x80;               //Red LED -> OFF
//        (*(volatile unsigned char *) 0x0021) &= ~0x80;
        for (i = 0; i < 10000; i++)
            ;   //delay
    }
}
```

Blink led example using direct write to port location

Exp02_LunchBox_HelloBlink_without_msp430_h.c

```c
//#include <msp430.h>
/*@brief entry point for the code*/
void main(void)
{
//  WDTCTL = WDTPW | WDTHOLD;
    (*(volatile unsigned int *) 0x0120) = 0x5A00 | 0x0080;
    /* ! Stop Watchdog (Not recommended for code in production
     and devices working in field)
     */

    //P1DIR |= BIT7;
    (*(volatile unsigned char *) 0x0022) |= 0x80;
    // P1.7 (Red LED)

    volatile unsigned long i;
    while (1)
    {

        //P1OUT |= BIT7;
//        P1OUT |= 0x80;                //Red LED -> ON
        (*(volatile unsigned char *) 0x0021) |= 0x80;
        //Red LED -> ON
        for (i = 0; i < 10000; i++)
            ;   //delay

        //P1OUT &=~ BIT7;
        //P1OUT &= ~0x80;               //Red LED -> OFF
        (*(volatile unsigned char *) 0x0021) &= ~0x80;
        for (i = 0; i < 10000; i++)
            ;   //delay
    }
}
```

Thank you!