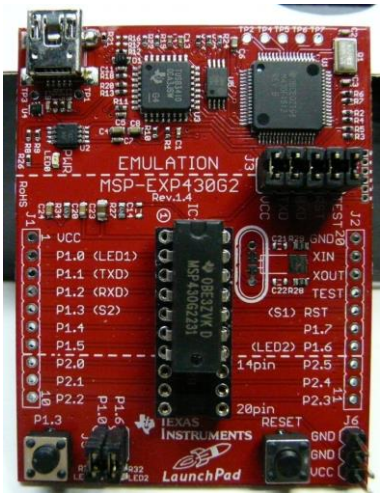
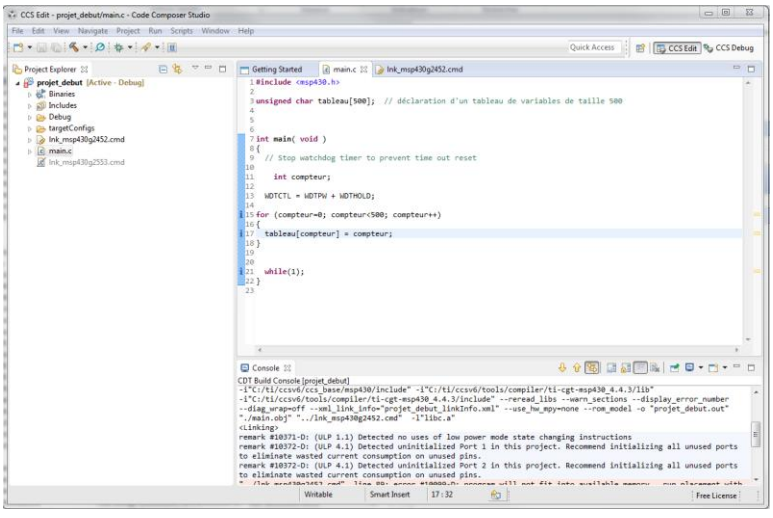
ESIGELEC	**LIVRET** **Systèmes à Microprocesseurs**	
2015-2016		**Département SEI**

Utilisation des fonctions de base d'un microcontrôleur

SEANCE n° 1

UTILISATION DE L'OUTIL DE DEVELOPPEMENT
ET
DE BIBLIOTHEQUES DE FONCTIONS





Règles de fonctionnement :

1. **Vous travaillez en équipe de 5 ou 6 étudiants définies par le tuteur.**

2. **Les séances durent 4 heures, toujours organisées en 2h tutorées (TUT) suivies de 2h en autonomie (A.P.P.)**

3. **La partie autonomie ne signifie pas ailleurs (le tuteur peut revenir faire l'appel)**

4. **A chaque séance, vous aurez une série de petits exercices à réaliser pour la séance suivante. Ces exercices serviront de base à des interrogations orales de certains d'entre vous au début de la séance suivante.**

5. **Copier sans comprendre ne sert strictement à rien. Le jour de l'examen de validation vous serez seul.**

Objectifs de la séance :

1. **Maîtriser l'utilisation de base de l'outil de développement et de la carte microcontrôleur.**

2. **Mettre en application les éléments de langage C vus précédemment.**

3. **Etre capable de conceptualiser et de réaliser un petit programme utilisant des éléments logiciels préexistants (bibliothèque de fonctions)**

4. **Vérifier et éventuellement corriger le fonctionnement d'une petite carte électronique.**

5. **Appréhender certains concepts techniques liés au développement sur microcontrôleur.**

Déroulement des séances :

30 min
oral d'élèves sur les exercices de la séance précédente
(ne concerne pas la première séance)

30 min
cours/exemple sur la thématique de la séance

30 min
exercice d'application directe à réaliser

30 min
correction de l'exercice précédent et
explication des exercices demandés pour la suite

2h
en autonomie, réalisation d'exercices par les équipes

Ressources

Vous disposez de plusieurs ressources matérielles, logicielles et humaines.

Ressources matérielles :
carte *Launchpad* MSP430, carte d'extension, module afficheur

Ressources logicielles :
environnement de développement *Code Composer Studio*

http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v6#Download

La version téléchargeable s'installe automatiquement avec une licence gratuite limitée à 16Koctets de code, ce qui est largement suffisant pour les exercices à réaliser.

Ressources humaines :
tuteurs et consultation d'expertise (en travail individuel)

Références documentaires

Carte MSP430

Toute la documentation propre à la carte Launchpad et au MSP430 est disponible sur l'ENT Systèmes à Microprocesseurs.

Langage C

Les liens suivants peuvent être utilisés pour s'initier sur le MSP430, le langage C et/ou C embarqué :

- http://www-ipst.u-strasbg.fr/pat/program/tpc.htm
-
- Langage C pour les systèmes embarqués :
  http://kadionik.developpez.com/cours/systeme/langage-c-embarque/

- http://electronique.marcel.free.fr/MSP430%20Ressources%20et%20petits%20programmes.html

- http://poujouly.net/ressources-techniques/msp430/

- http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_Tutorials (le site de référence sur la Launchpad)

TI LaunchPad

- Get started with microcontroller LaunchPad Evaluation Kits from Texas Instruments.
- Choose from a variety of low-cost kits & BoosterPack plug-in modules.
- Scalable software tools provide multiple points of entry for programming your LaunchPad.

MSP-EXP430G2

On-board emulation

The MSP430 LaunchPad features on-board emulation, which means you can program and debug your projects without the need for additional tools.

BoosterPack-compatibility

All pins of the MSP430G2 device are fanned out for easy access. These pins make it easy to plug in 20-pin BoosterPacks that add additional functionality like wireless, capacitive touch and more

Switches & LEDs

A general purpose switch, reset switch and two LEDs are provided on-board for quick development.

What's included?

MSP430 LaunchPad includes everything you need to get started with MSP430 development!

- MSP-EXP430G2 LaunchPad
- MSP430G2452 microcontroller
- Mini-USB cable
- Quick Start Guide
- 32kHz external crystal

Project Zero: Blink your first LED

My First MSP430 LaunchPad Project

In this project, we will learn a few things:

- How to create a new project with Code Composer Studio
- Learn how to blink the on-board Red LED on the MSP430 LaunchPad
- Change the speed of the blinking Red LED
- Learn how to toggle between the Red and Green LED

Things you will need

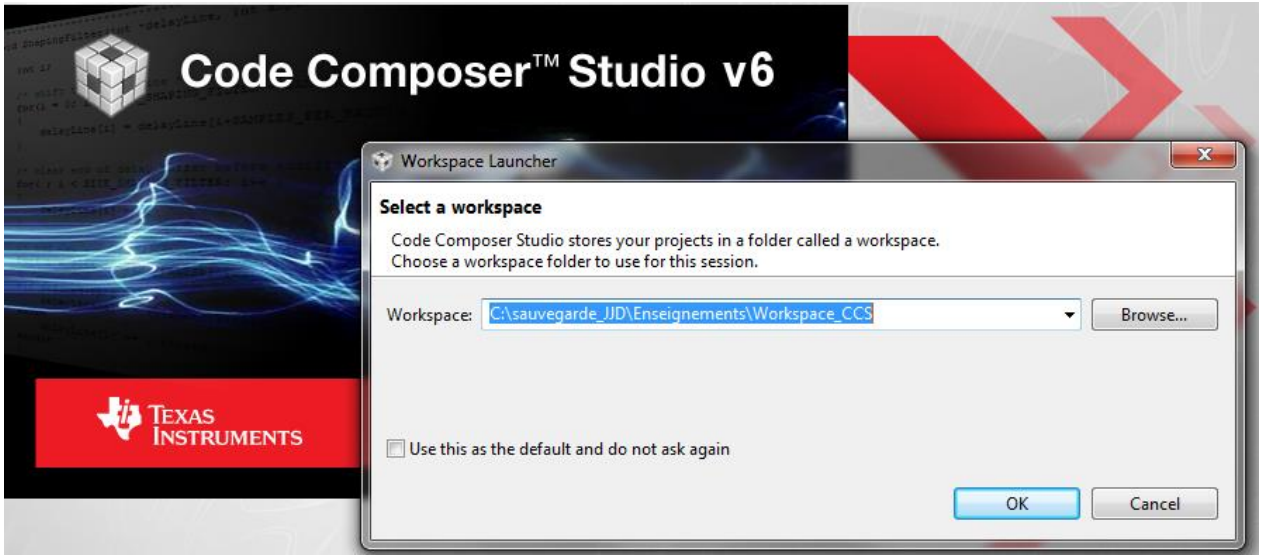
1. MSP430 LaunchPad Evaluation Kit (MSP-EXP430G2)
2. Code Composer Studio (Software Development Environment)
3. 10 minutes

Hardware setup

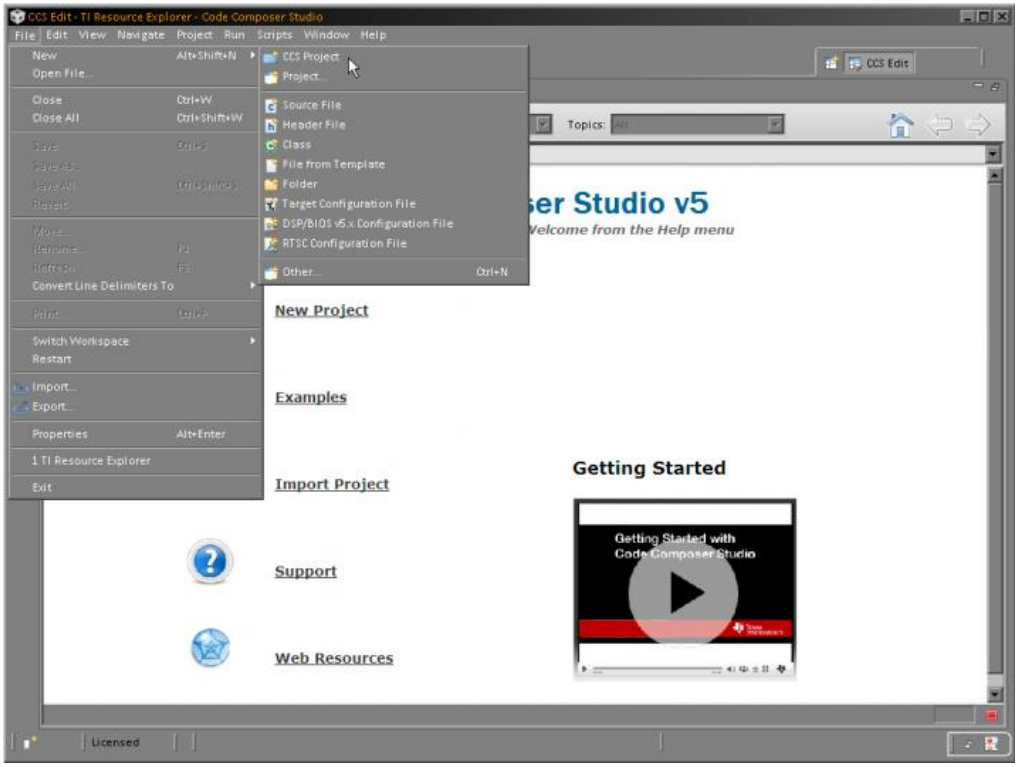
1. The LaunchPad kit includes everything you need out of the box
2. Plug your LaunchPad into the PC with the included USB cable
3. If prompted, let Windows automatically install the software.

Create a new CCS workspace

1. Upon opening CCS, it will ask you to select a workspace
2. Since this is our first project, we'll create a new one called "LaunchPad_Projects". A workspace is where all of your Code Composer Studio projects will live. Once created, press OK

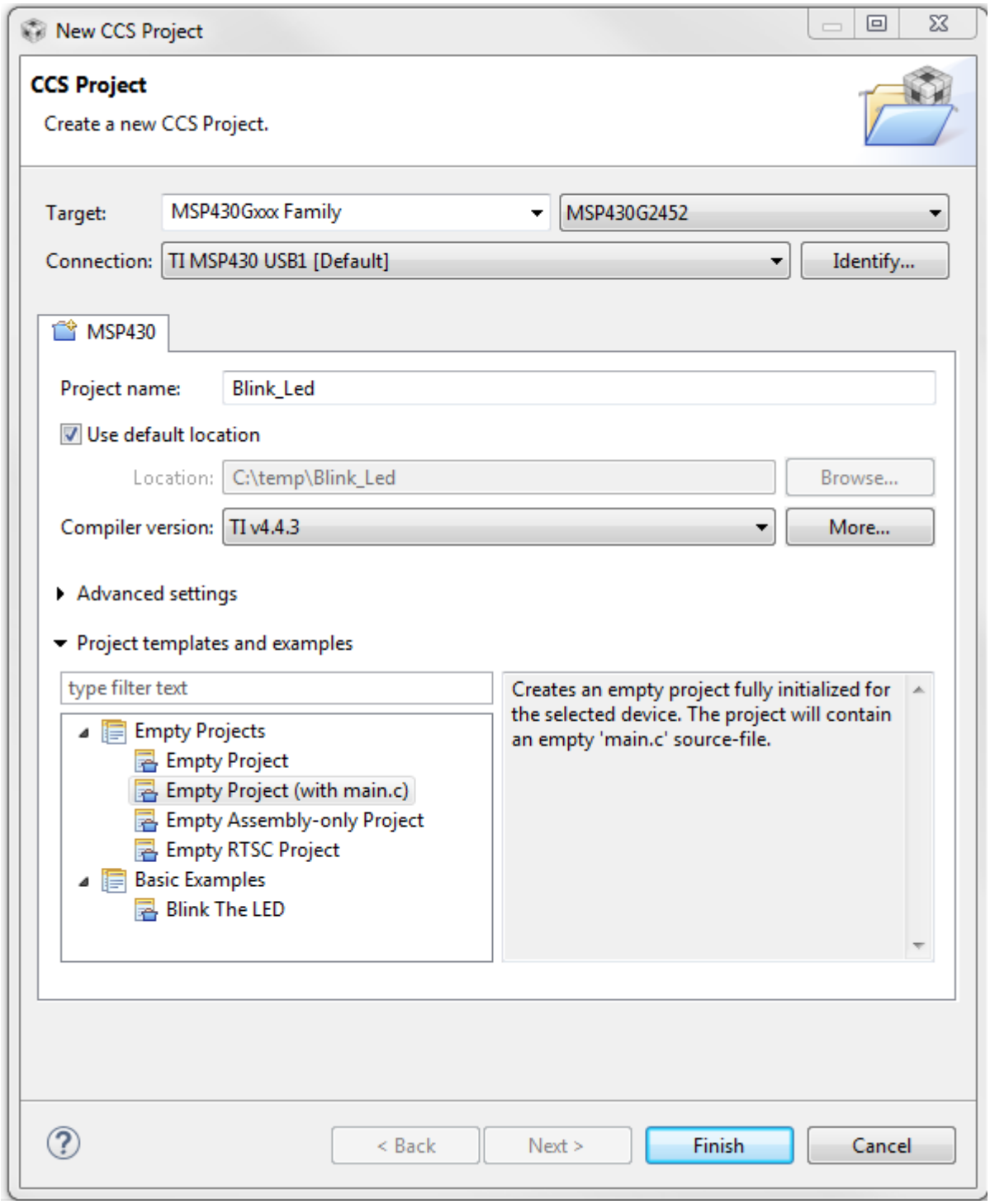


3. Once CCS is opened, we can create a new project by going to File > New > CCS Project



4. This will open up the "New CCS Project" Window. Within this window, we need to do 2 things. Name our project & choose our Device Variant. Let's name our project "Blink_LED"

   We also need to choose the appropriate MSP430 device. For this tutorial, we will program the MSP430G2452 device that comes pre-populated on the MSP430 LaunchPad. (Due to the simplicity of this particular tutorial, any of the MSP430G2xx devices will work for this example!) Then, click "Finish"

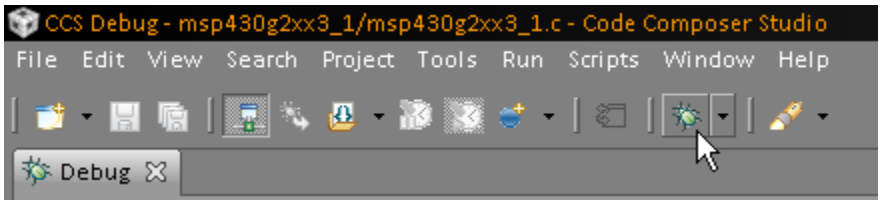

New CCS Project
CCS Project
Create a new CCS Project.
Target:
MSP430Gxxx Family
MSP430G2452
Connection:
TI MSP430 USB1 [Default]
Identify...
MSP430
Project name:
Blink_Led
Use default location
Location:
C:\temp\Blink_Led
Browse...
Compiler version:
TI v4.4.3
More...
Advanced settings
Project templates and examples
type filter text
Empty Projects
Empty Project
Empty Project (with main.c)
Empty Assembly-only Project
Empty RTSC Project
Basic Examples
Blink The LED
Creates an empty project fully initialized for the selected device. The project will contain an empty 'main.c' source-file.
< Back
Next >
Finish
Cancel

Writing code

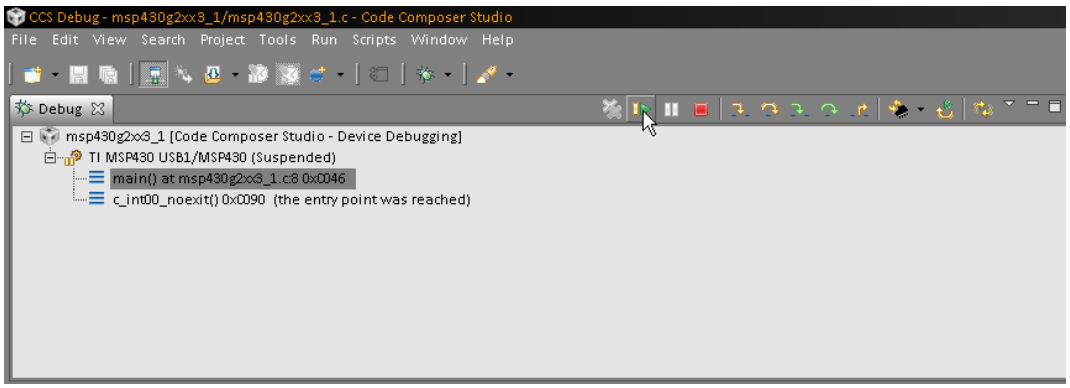
Now, we have our blank canvas! We can finally start writing code!

```
1.  #include <msp430.h>
2.   unsigned int i = 0;
3.
4.       // Initialize variables. This will keep count of how many cycles between
   LED toggles
5.
6.
7.   void main(void)
8.   {
9.       WDTCTL = WDTPW + WDTHOLD;
10.
11.      // Stop watchdog timer. This line of code is needed at the beginning of
   most MSP430 projects.
12.
13.      // This line of code turns off the watchdog timer, which can reset the
   device after a certain period of time.
14.      // P1DIR is a register that configures the direction (DIR) of a port pin as
   an output or an input.
15.
16.      P1DIR |= 0x01;
17.
18.
19.      // To set a specific pin as output or input, we write a '1' or '0' on the
   appropriate bit of the register.
20.
21.
22.      // P1DIR = <PIN7><PIN6><PIN5><PIN4><PIN3><PIN2><PIN1><PIN0>
23.
24.
25.      // Since we want to blink the on-board red LED, we want to set the
   direction of Port 1, Pin 0 (P1.0) as an output
26.
27.      // We do that by writing a 1 on the PIN0 bit of the P1DIR register
28.      // P1DIR = <PIN7><PIN6><PIN5><PIN4><PIN3><PIN2><PIN1><PIN0>
29.      // P1DIR = 0000 0001
30.      // P1DIR = 0x01     <-- this is the hexadecimal conversion of 0000 0001
31.
32.
33.      for (;;)
34.
35.      // This empty for-loop will cause the lines of code within to loop
   infinitely
36.
37.          {
38.
39.              P1OUT ^= 0x01;
40.
41.      // Toggle P1.0 using exclusive-OR operation (^=)
42.
43.      // P1OUT is another register which holds the status of the LED.
44.      // '1' specifies that it's ON or HIGH, '0' specifies that it's OFF or LOW
45.      // Since our LED is tied to P1.0, we will toggle the 0 bit of the P1OUT
   register
46.
47.              for (i=20000; i>=1; i--);
48.   //Delay between LED toggles. This for-loop will run until the condition is
   met.
49.   //In this case, it will loop until the variable i decrements from 20000 to 0.
50.          }
51.      }
```

1. Now that we have written our code, we can download it to our MSP430 LaunchPad that is plugged into the USB port! We can do this by clicking the debug button



2. Clicking the Debug button will take us to the CCS Debug View. Here, we can press the Run button to start running the code we just wrote.



3. At this point, your Red LED should be blinking! Congratulations!

Other exercises

1. Now that we have our LED blinking, play around with the number inside of the for-loop to change the speed of the blinking LED. The smaller the number, the shorter the delay between LED toggles. Alternatively, the larger the number, the longer the delay. Try values like 5000, 40000, etc.

2. Another exercise is getting the green LED to blink as well. The green LED is tied to Port P1.6. Using the P1DIR and P1OUT registers we used above, see if you can get both LEDs to blink. Can you make them blink in unison? Can you make them blink alternatively?