Malaga University
Electronic Technology Department

ETS Telecommunications Engineering

Bachelor's Degree in Telecommunications Systems Engineering

microcontrollers

Course 2022/2023

Practice 1: CCS Tutorial

(v2)

INDEX

1. INTRODUCTION

This document describes **Practice 1** of the course, which presents the *Code Composer Studio* (CCS) development environment used in it for the implementation of applications based on microcontrollers.

The *Code Composer Studio* development environment has been created by *Texas Instruments,* the manufacturer of the MSP430G2533 microcontroller used in the course, and can be downloaded directly from their Web:

https://www.ti.com/tool/CCSTUDIO

Although you can download the latest version of the development environment available, it is recommended for compatibility to use the same one that is installed in the laboratory (11.1.0.00011 of December 21, 2021), which can be obtained from the Texas Instruments website itself . :

https://software-dl.ti.com/ccs/esd/CCSv11/CCS_11_1_0/exports/CCS11.1.0.00011_win64.zip

It is also recommended to use the *offline* version of the installer, and during the installation process select only the *MSP430 ultra-low power MCUs* option to reduce the size of the installation.

To work in the laboratory, the directory **"D:\USR\uC\Gxx"** 1 will be used for the , where you have to replace "Gxx" degree in which the subject is studied (GTT, GST, GSE, GSI, GTm). The workspace, projects and source code files of the different practices implemented in the subject will be stored in this directory. To make a backup copy2 of the work carried out in the laboratory, it is enough to copy said directory.

Finally, it is important to note that this tutorial is intended to be a simplified guide to using the *Code Composer Studio* development environment available in the laboratory, so depending on the version installed, some options and menus may differ slightly from those here described.
In addition, as it is a simplified guide, it does not intend to describe all the options available in the Code Composer Studio development environment , but only those essential options to start developing applications based on the microcontroller used in the course.

[1] If the directory does not exist, it must be created.

[2] **VERY IMPORTANT:** The laboratory is a space shared by numerous students from different subjects and degrees, so the files stored on each computer can be modified and/or deleted at any time. It is therefore recommended to copy the work done in each session to be able to restore it in the next session.

2. CODE COMPOSER STUDIO

The *Code Composer Studio* (CCS) tool constitutes an Integrated Development Environment or IDE *(Integrated Development Environment),* which is an application that incorporates all the necessary tools to centrally carry out the complete flow of *software* development of any application based on microcontrollers (editing, compiling, executing, debugging…).

These tools are distributed in two basic components:

– **Editor:** allows you to edit and compile source code.

– **Debugger:** Allows you to run and debug executable code.

Each of these components has its own graphical interface (menus, toolbars...), and you can switch between one and the other at any time[3] .

Code Composer Studio is hierarchically structured into **workspaces** and **projects:**

- Workspaces

  Workspaces group all the projects related to each other in the implementation of a certain application. You can have as many workspaces as you want, but you can only work with one of them at a time.

  A workspace can contain as many projects as you want, but you can only work with one of them at a time, which is called the active project[4] . All the operations that are carried out (assembly/compilation, linking, debugging...) are carried out on the active project[5] .

- *Projects*

  The projects group all the files (both the code files and other types of files necessary to assemble/ compile and link the project) that are used together to implement a certain application, which will be assembled/compiled and linked together to create a program. executable.

[3] To be able to access the debugger it is necessary to have compiled code.

[4] The active project is recognized by its name appearing in bold in the project browser together with the label “Active”.

[5] The active project in the workspace can be changed by selecting the desired project in the project explorer.

Each project has its own configuration options, among which are, for example, the type of microcontroller for which the code is developed or the programming language used (assembler or C).

Thus, an appropriate structure for working on the subject consists of declaring a single workspace6 (for example, "uC"), and within it, creating a project for each of the practices developed in the subject (for example, " p2.1", "p2.2", "p3.1"…).

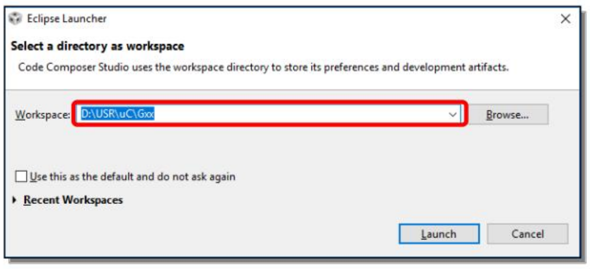
[6] This philosophy also allows you to quickly make copies of all the work done, since all the information used by *Code Composer Studio* is stored in the directory associated with the workspace. Thus, by copying said directory you can make a backup copy of all the projects, configurations, files... developed.

3. PUBLISHER

The first step in implementing microcontroller-based applications is to edit the corresponding source code. For this, the *Code Composer Studio* editor is used , following the steps indicated below.

3.1. initialization

When running *Code Composer Studio* it requests the directory[7] where the workspace is to be located .


Eclipse Launcher

Select a directory as workspace

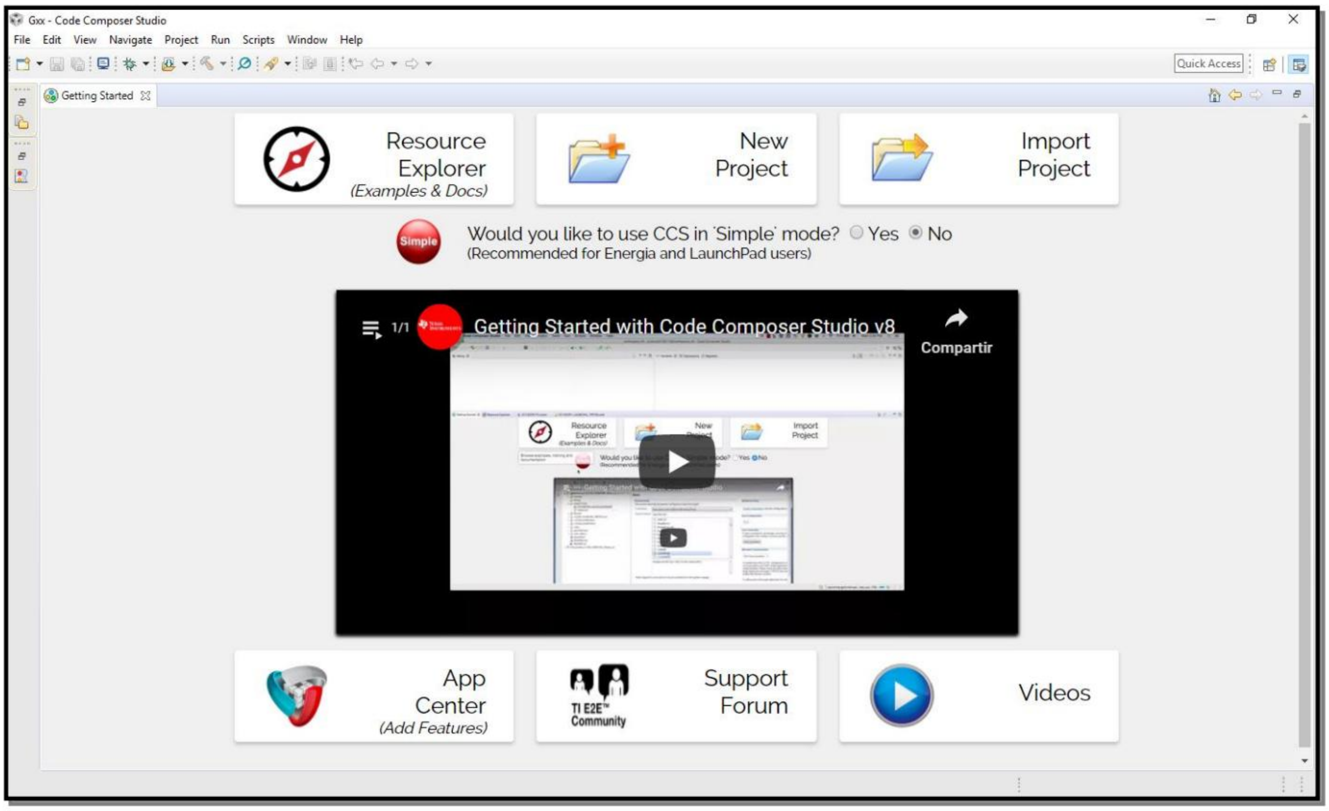
Code Composer Studio uses the workspace directory to store its preferences and development artifacts.

Workspace: D:\USR\uC\Gxx

☐ Use this as the default and do not ask again

Recent Workspaces

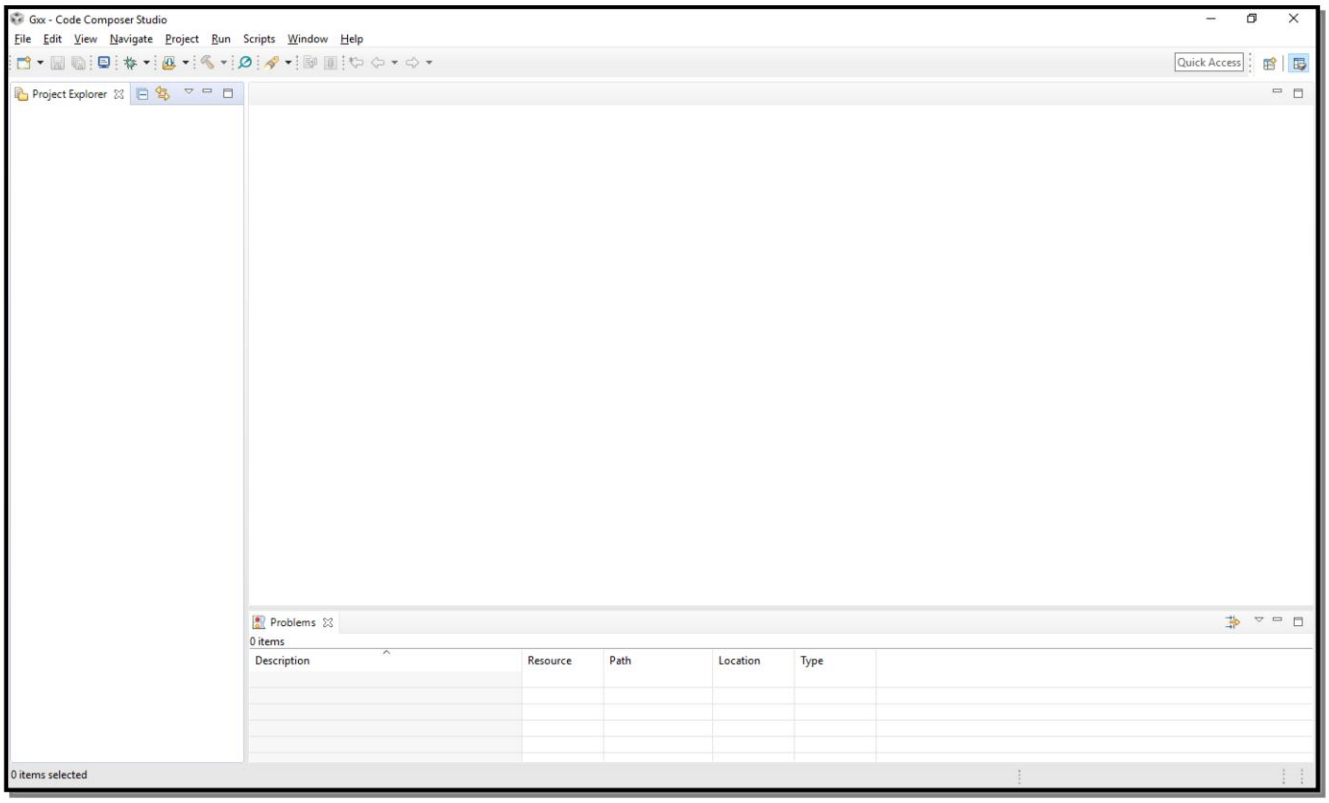
After selecting the directory, the main editor window opens, which initially shows a welcome screen.


Gxx - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Getting Started
Resource Explorer (Examples & Docs)
New Project
Import Project
Simple
Getting Started with Code Composer Studio v8
Compartir
App Center (Add Features)
TI E2E Community
Support Forum
Videos

Would you like to use CCS in 'Simple' mode? Yes [✓ No]
(Recommended for Energia and LaunchPad users)

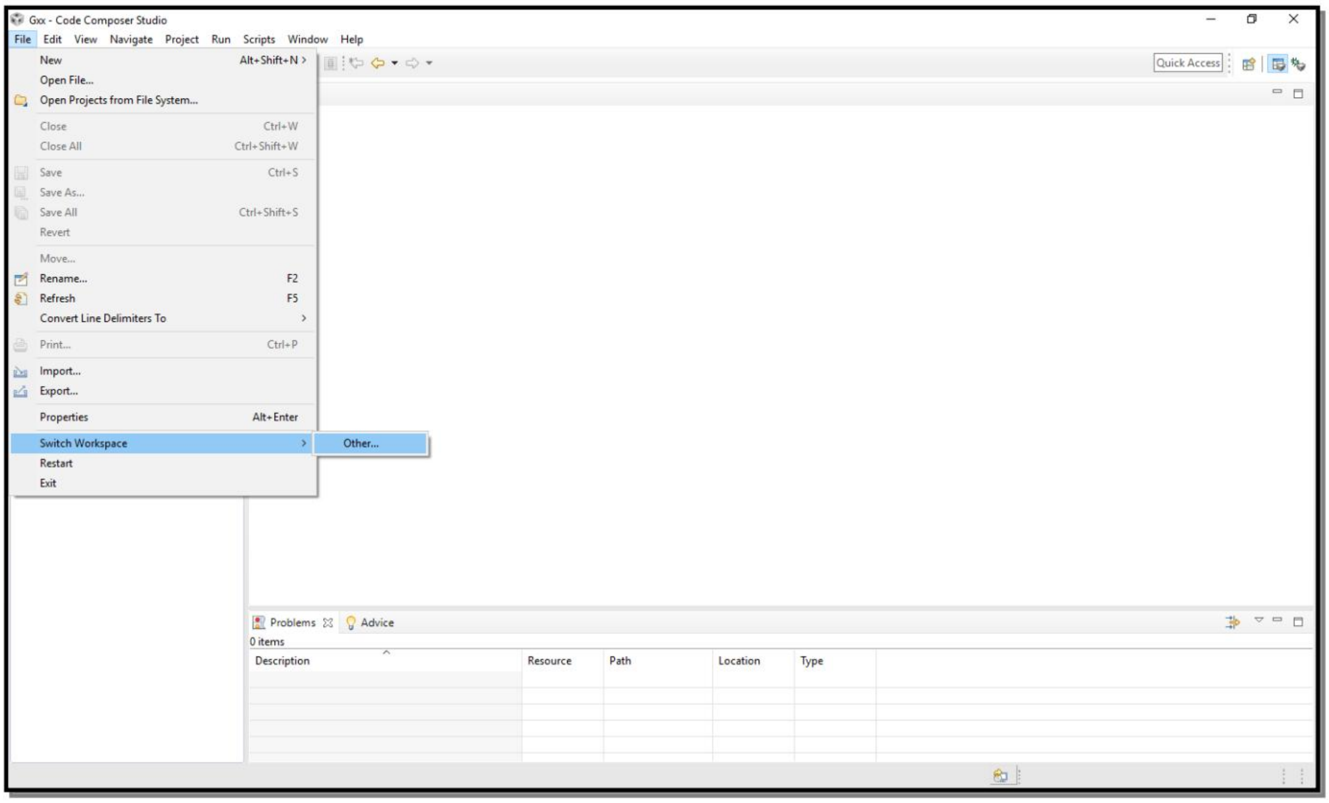
[7] As described in the introduction, the directory **"D:\USR\uC\Gxx" will be used,** where "Gxx" must be replaced by the degree in which the subject is studied (GTT, GST, GSE , GSI, GTm).

Once the welcome screen is closed, the editor is shown with its main work windows (project explorer, code editor and problems).



At any time you can switch to another workspace to work in with the option:

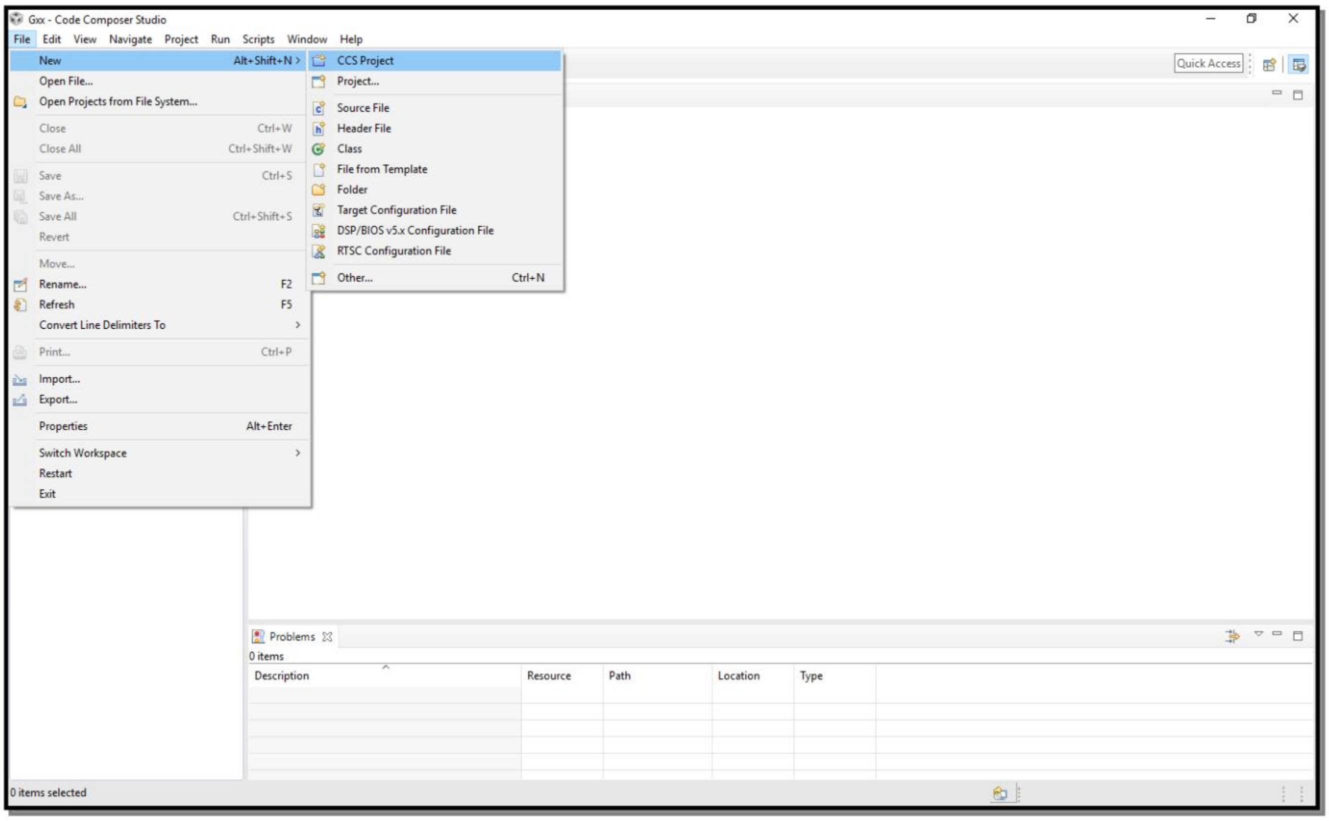
– File ÿ Switch Workspace ÿ Other…



3.2. Project creation and configuration

To start editing the source code, it is first necessary to create and configure a project in which to locate said source code, so that it can be correctly compiled. To do this, you must choose the option:

– File ÿ New ÿ CCS Project

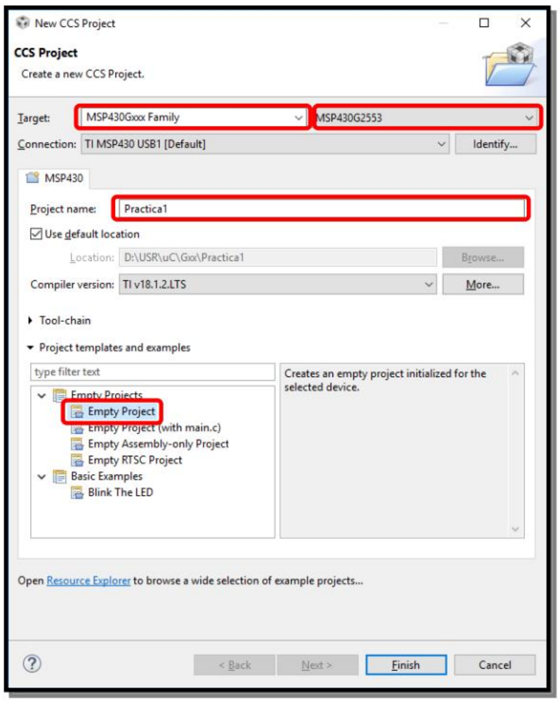


and select the following options to configure the project:

– Target: MSP430Gxxx Family ÿ MSP430G2553

– Project Name: Practice1

– Project Template: Empty Project



New CCS Project

CCS Project

Create a new CCS Project.

Target: MSP430Gxxx Family | MSP430G2553

Connection: TI MSP430 USB1 [Default] | Identify...

MSP430

Project name: Practica1

☑ Use default location

Location: D:\USR\uC\Gxx\Practica1 | Browse...

Compiler version: TI v18.1.2.LTS | More...

Tool-chain

Project templates and examples

type filter text

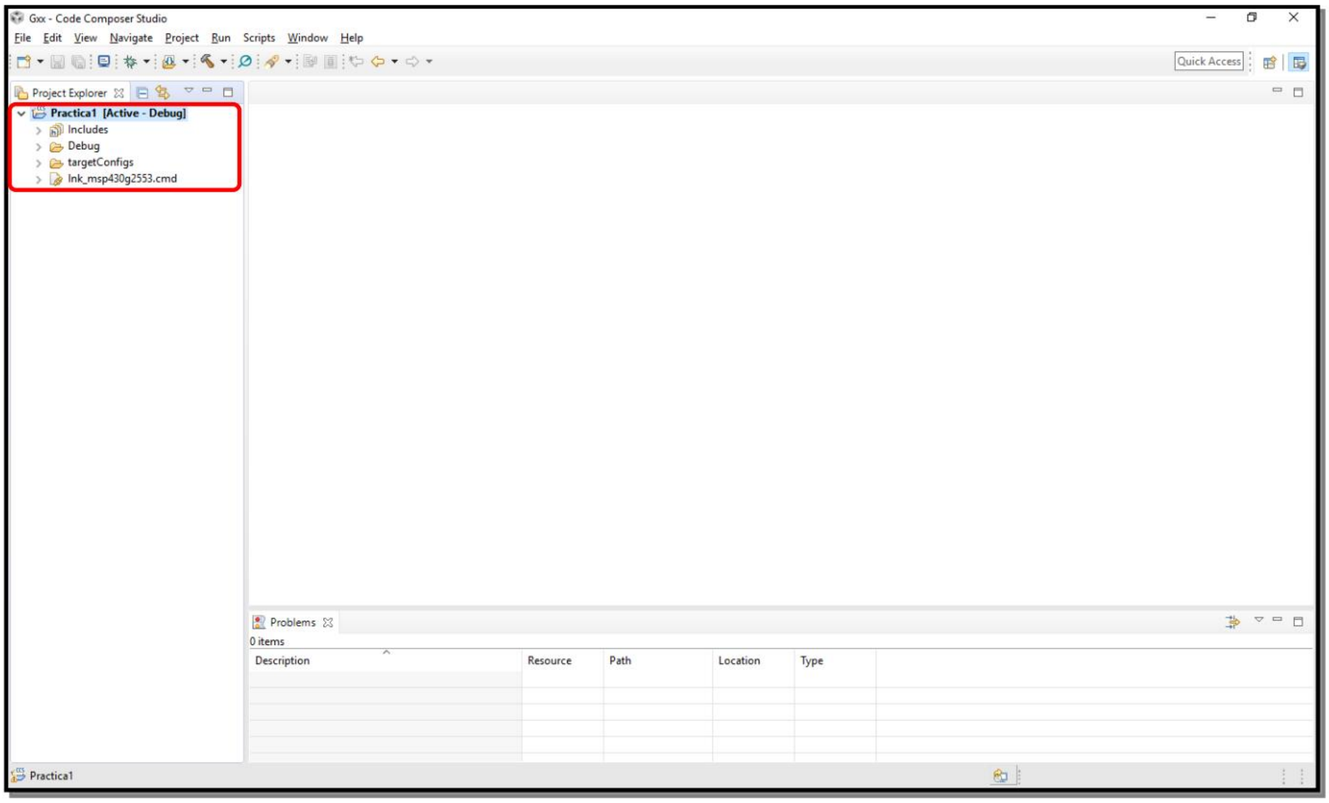
- Empty Projects
  - Empty Project
  - Empty Project (with main.c)
  - Empty Assembly-only Project
  - Empty RTSC Project
- Basic Examples
  - Blink The LED

Creates an empty project initialized for the selected device.

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

< Back | Next > | Finish | Cancel

Once created and configured the project will appear in the project explorer.



Gxx - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Quick Access

Project Explorer

- Practica1 [Active - Debug]
  - Includes
  - Debug
  - targetConfigs
  - lnk_msp430g2553.cmd

Problems

0 items

Description	Resource	Path	Location	Type

Practica1

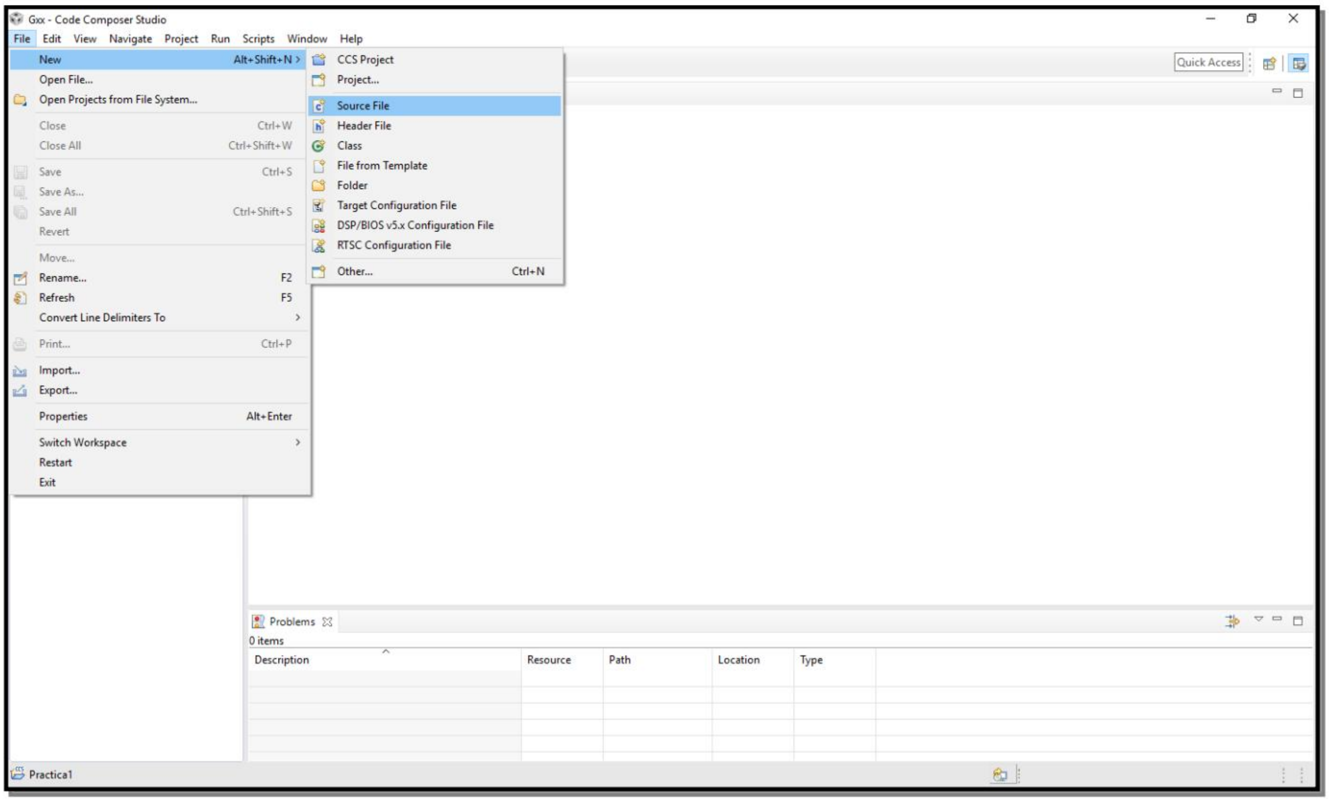
3.3. Editing the source code

After creating and configuring the project, you have to edit the source code to be developed. There are two options for carry out this task:

3.3.1. new file

If you want to create the source code from the beginning in a new file, you must select the option:

– File ÿ New ÿ Source File

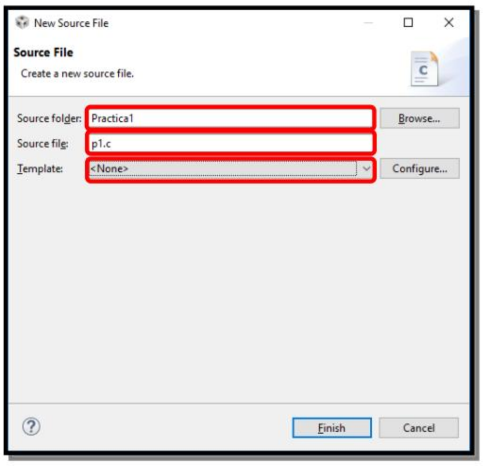


and select the following options to create the file:

– Source folder: Practice1

– Source file: p1.c

–Template: None


New Source File

Source File

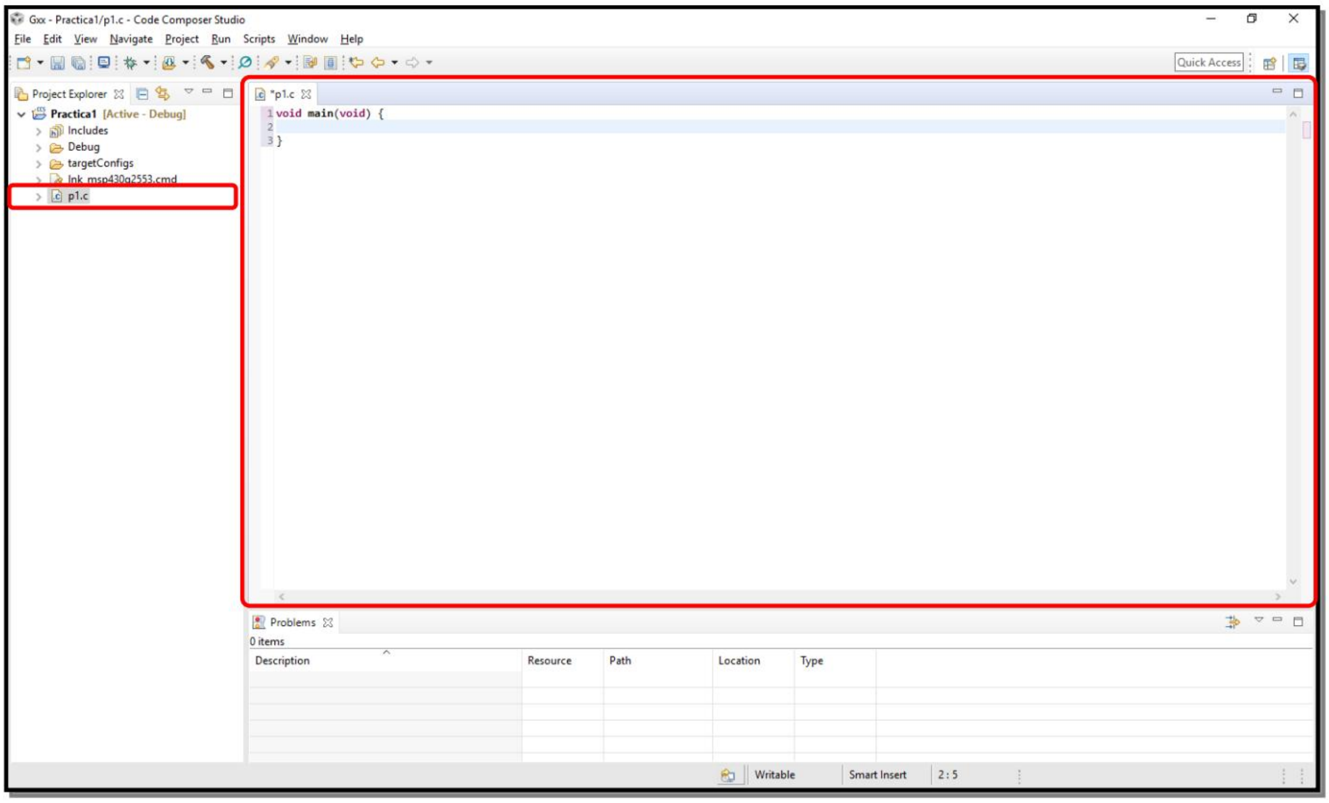
Create a new source file.

Source folder: Practica1

Source file: p1.c

Template: <None>

Once the file has been created, it appears both in the project explorer and in the editor, making it possible to start editing it:



```
void main(void) {

}
```

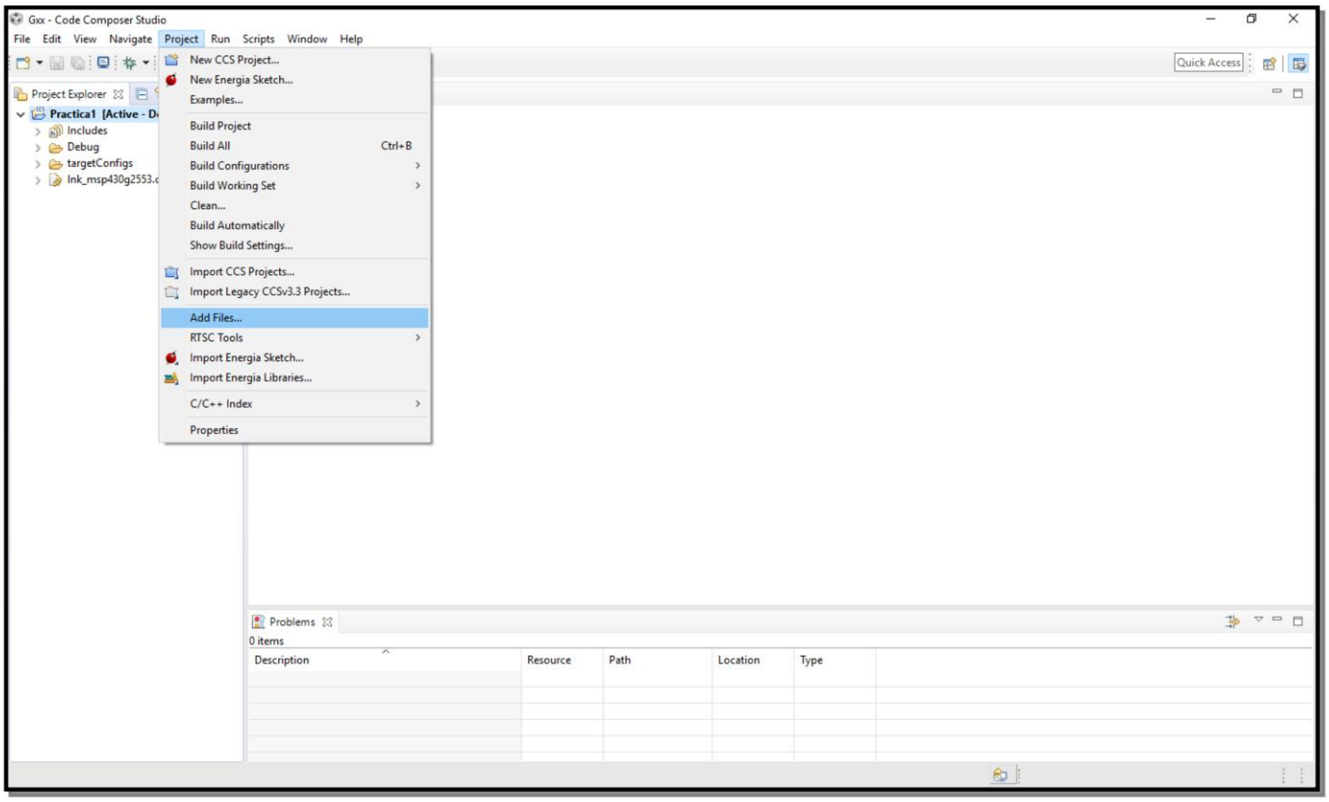
3.3.2. existing file

If you want to use an existing source code file[8] you simply have to incorporate it into the project using the option:

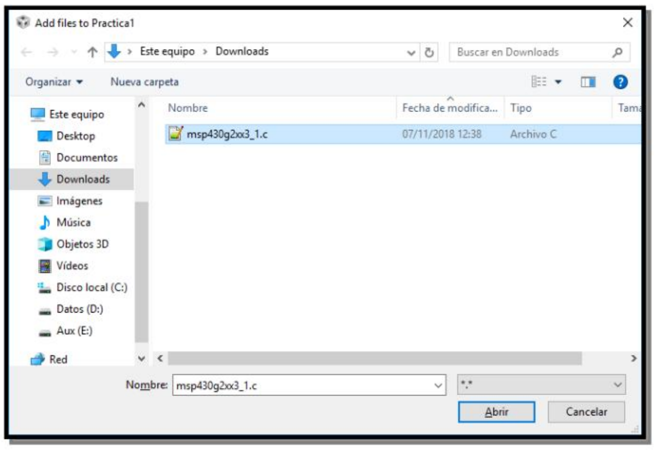
–Project ÿ Add Files…

[8] In order to show the complete code editing and debugging process of the *Code Composer Studio* development environment , the file "msp430g2xx3_1.c" will be used as source code, which corresponds to one of the examples of use of the microcontroller provided by *Texas Instruments.* . Both the examples of use and the file "msp430g2xx3_1.c" are available on the Virtual Campus of the subject.

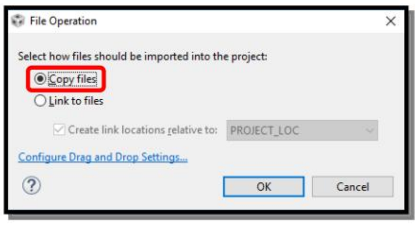
Machine Translated by Google



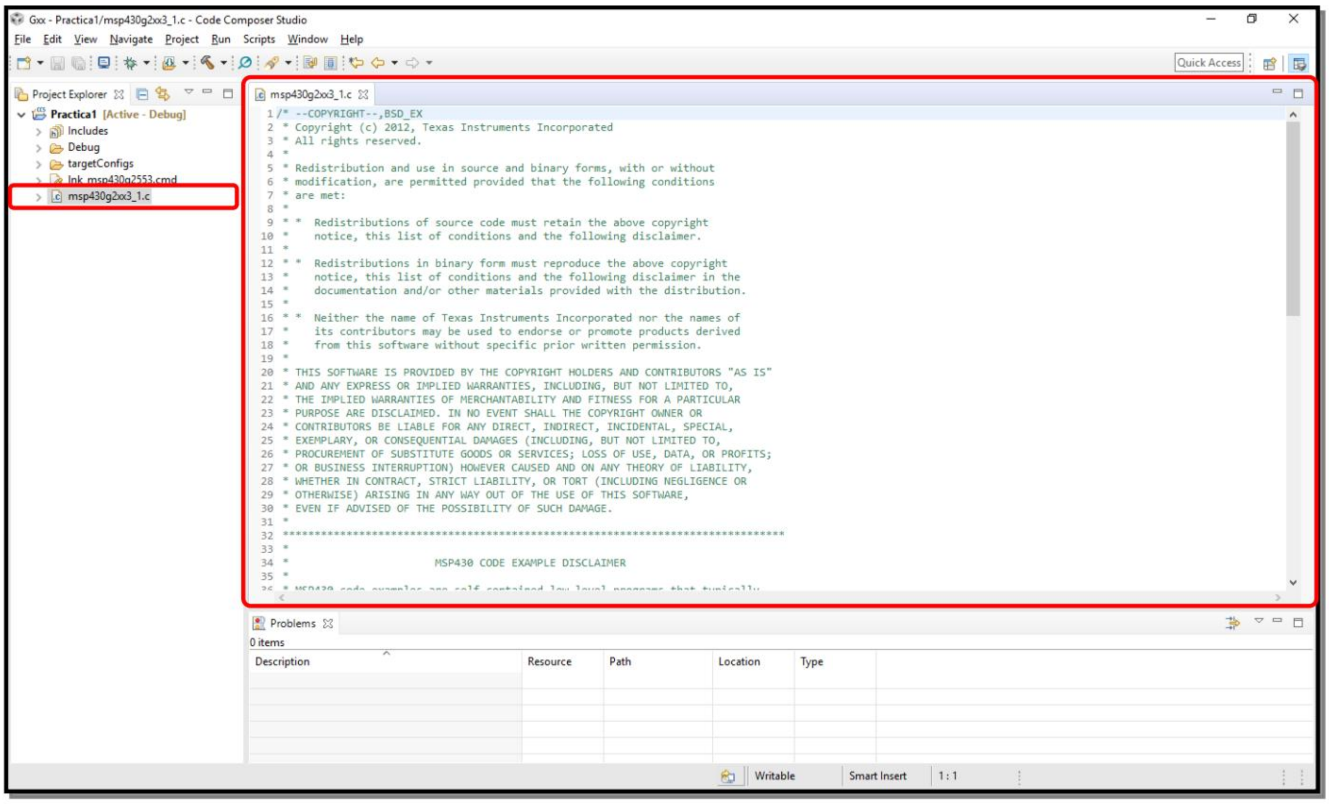
and select the file that you want to include in the project:



indicating to copy the file into the workspace:


File Operation
Select how files should be imported into the project:
Copy files
Link to files
Create link locations relative to:
PROJECT_LOC
Configure Drag and Drop Settings...
OK
Cancel

After carrying out this process, the file appears in the project explorer, and double- *clicking* on it opens it in the editor:



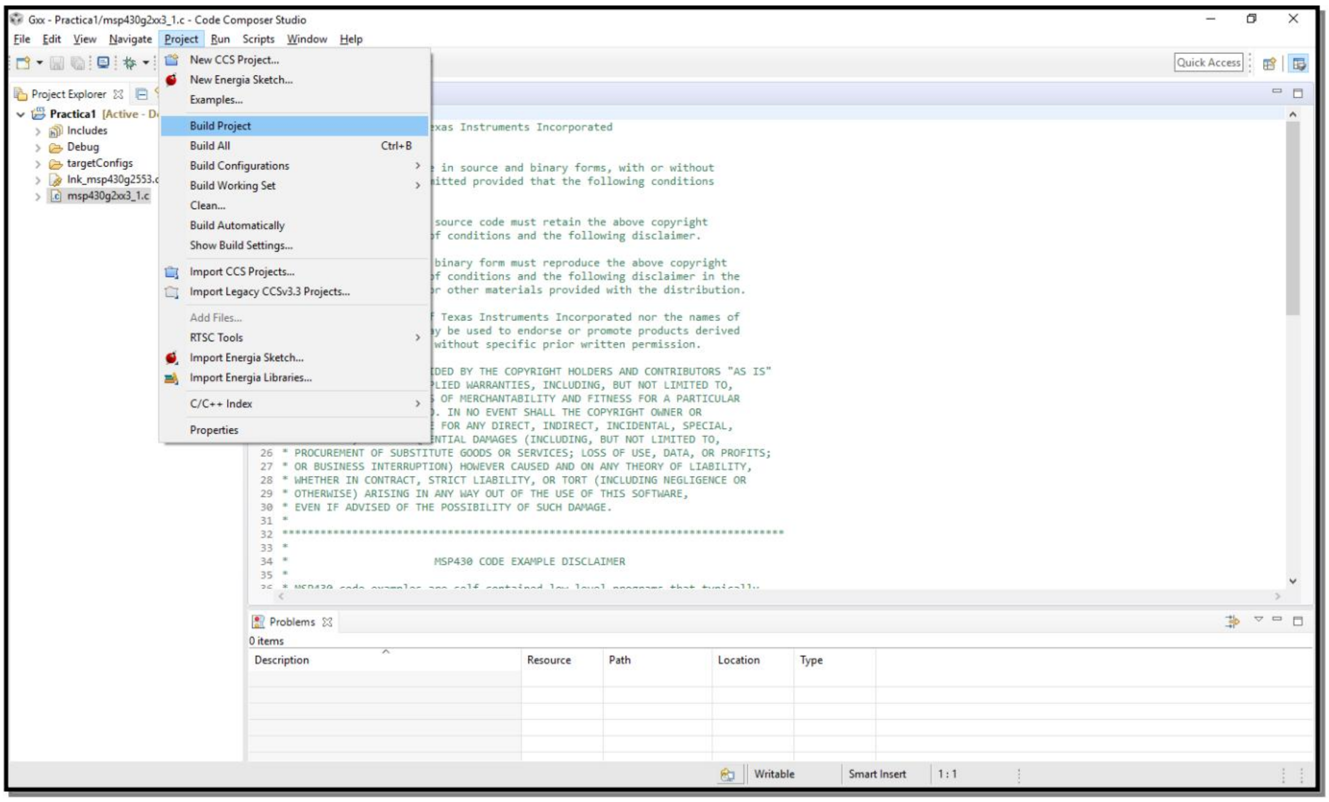
```
/* --COPYRIGHT--,BSD_EX
 * Copyright (c) 2012, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 *
 *                       MSP430 CODE EXAMPLE DISCLAIMER
 *
```

3.4. Compilation

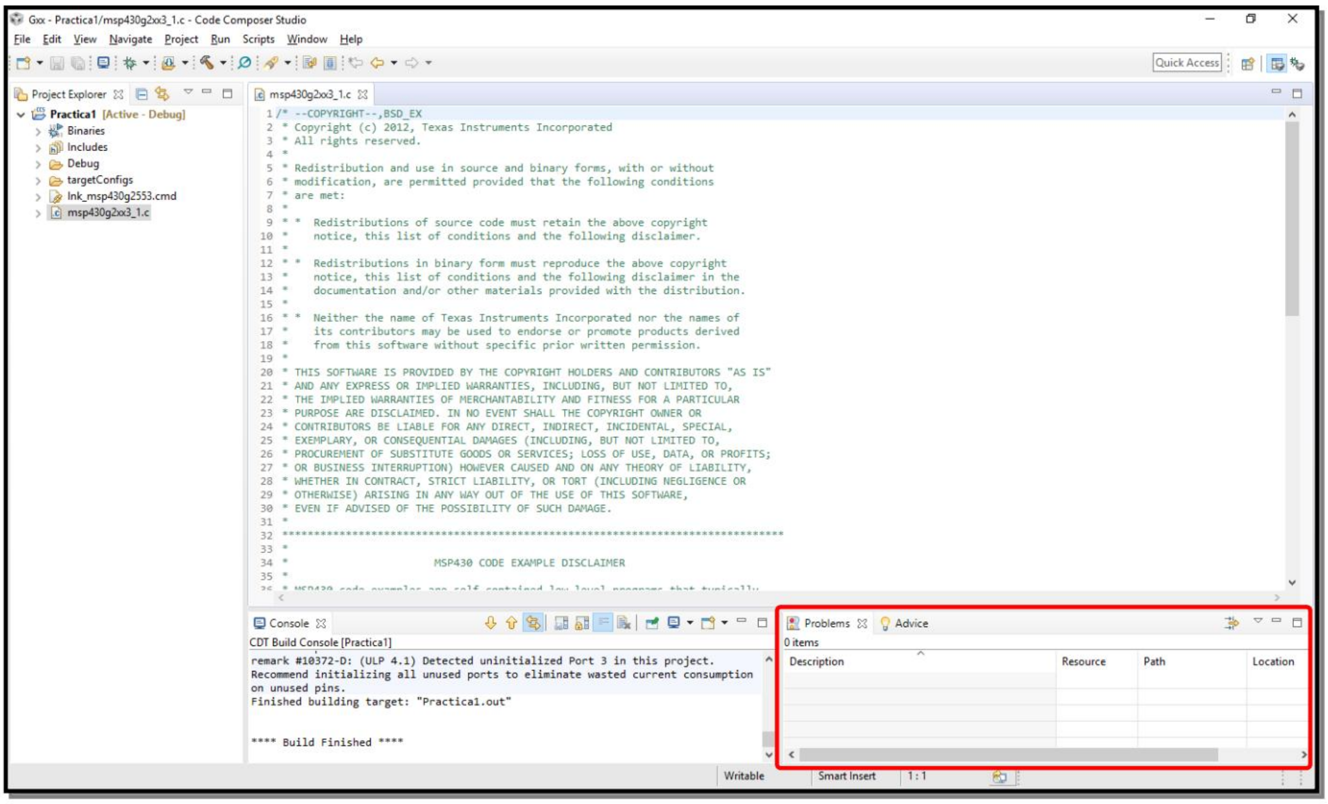
Once the source code[9] to be developed is available, it is necessary to compile it to generate an executable code. To do this, you must choose the option:

–Project ÿ Build Project

[9] In order to show the complete code editing and debugging process of the *Code Composer Studio* development environment , the file "msp430g2xx3_1.c" will be used as source code, which corresponds to one of the examples of use of the microcontroller provided by *Texas Instruments.* . Both the examples of use and the file "msp430g2xx3_1.c" are available on the Virtual Campus of the subject.



If there are no errors in the source code, the compilation process is carried out, generating the corresponding executable code. If there are errors in the source code they are shown in the problems window.


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
Practica1 [Active - Debug]
Binaries
Includes
Debug
targetConfigs
lnk_msp430g2553.cmd
msp430g2xx3_1.c
msp430g2xx3_1.c

```
/* --COPYRIGHT--,BSD_EX
 * Copyright (c) 2012, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 *
 *                       MSP430 CODE EXAMPLE DISCLAIMER
 *
```

```
remark #10372-D: (ULP 4.1) Detected uninitialized Port 3 in this project.
Recommend initializing all unused ports to eliminate wasted current consumption
on unused pins.
Finished building target: "Practica1.out"


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

Problems
Advice
0 items
Description Resource Path Location
Writable Smart Insert 1 : 1

4. DEBUGGER

Once the executable code has been generated, its operation can be analyzed using the *Code Composer Studio debugger,* following the steps indicated below[10] .

VERY IMPORTANT: *Code Composer Studio* has a bug reported by *Texas Instruments* in code debugging, which causes a crash during debugging when the code uses interrupts (erroneously disables the GIE bit of the SR register under certain circumstances, so interrupts They stop working). To prevent this error from affecting code debugging, the following rules must be followed:

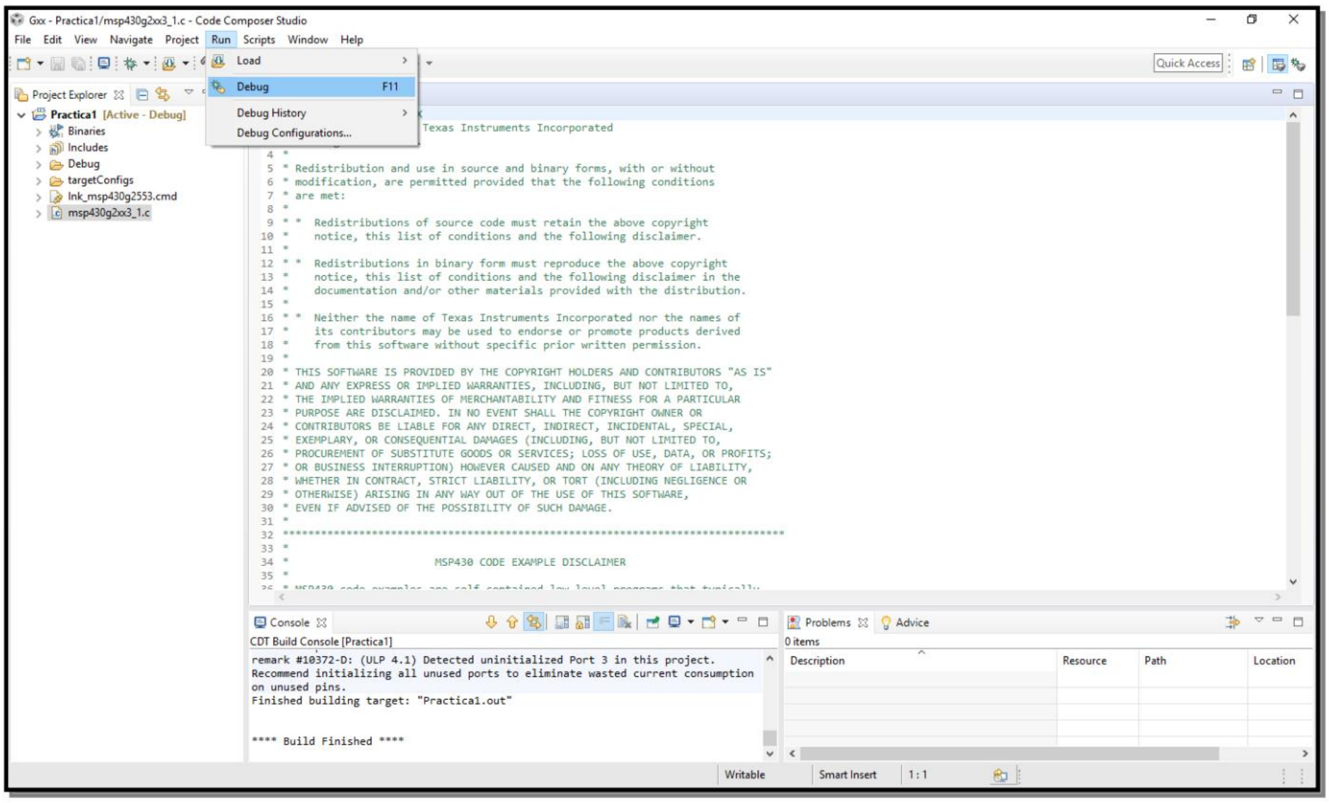
1. Only use **a breakpoint** when debugging code. If it is necessary to use several breakpoints, the breakpoint must be deactivated and activated throughout the code, guaranteeing that a single breakpoint is always maintained.

2. **Do not use functions** in the code (except for the RTIs, which can be used without problem).

4.1. initialization

To open the debugger you have to choose the option:

–Run ÿ Debug

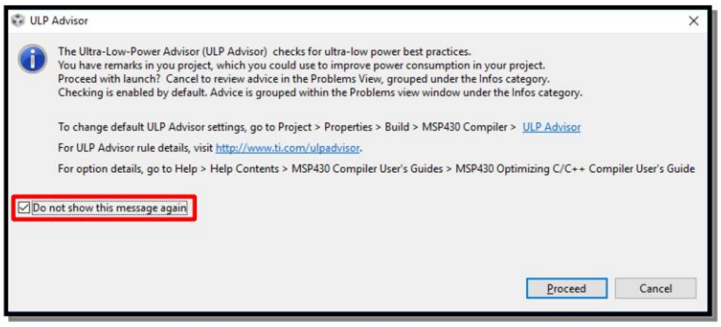


```
 Texas Instruments Incorporated
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 *
 *                       MSP430 CODE EXAMPLE DISCLAIMER
 *
```

```
CDT Build Console [Practica1]
remark #10372-D: (ULP 4.1) Detected uninitialized Port 3 in this project.
Recommend initializing all unused ports to eliminate wasted current consumption
on unused pins.
Finished building target: "Practica1.out"


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

[10] To start debugging it is necessary to transfer the executable code to the microcontroller, so you have to connect the Development Kit to the PC before you can continue.

An informative message may then appear reminding you that there are consumption optimizations pending analysis *(ULP Advisor).* You can ignore these optimizations and check the option not to.
show again:

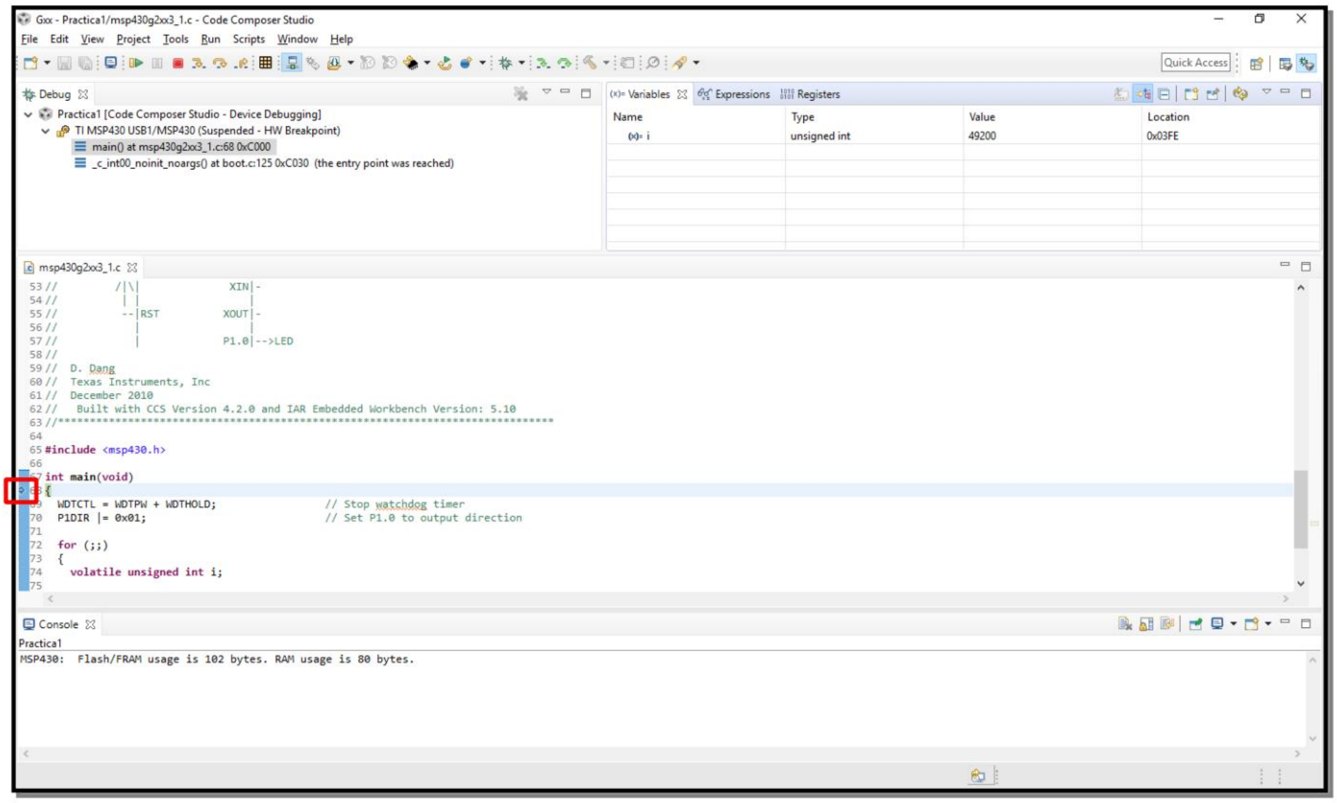

ULP Advisor

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.

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, go to Help > Help Contents > MSP430 Compiler User's Guides > MSP430 Optimizing C/C++ Compiler User's Guide

[✓] Do not show this message again

Finally, the executable code is transferred to the microcontroller, starting debugging. During the debugging process, the code statement to be executed next is always displayed, identified by the symbol


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio

Practica1 [Code Composer Studio - Device Debugging]
TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
main() at msp430g2xx3_1.c:68 0xC000
_c_int00_noinit_noargs() at boot.c:125 0xC030 (the entry point was reached)

Name	Type	Value	Location
i	unsigned int	49200	0x03FE

```
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|-
//            |                 |
//            |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

```

Practica1

```
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

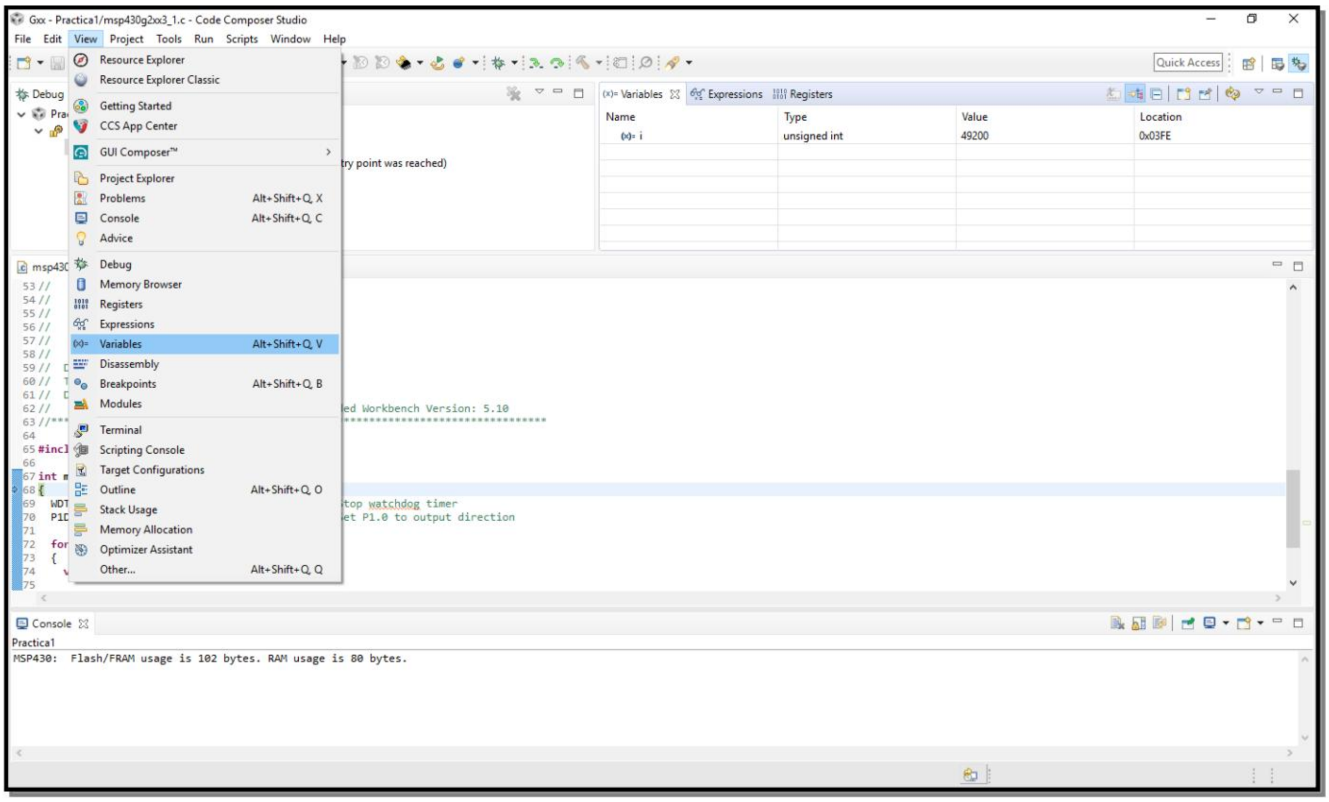
4.2. Display

From the debugger it is possible to view any microcontroller resource (code, variables, peripherals, memory...) at any time. The main visualization tools are described below.

4.2.1. variables

The variables window allows you to control the variables of the code. To show the variables window it may be necessary to enable it with the option:

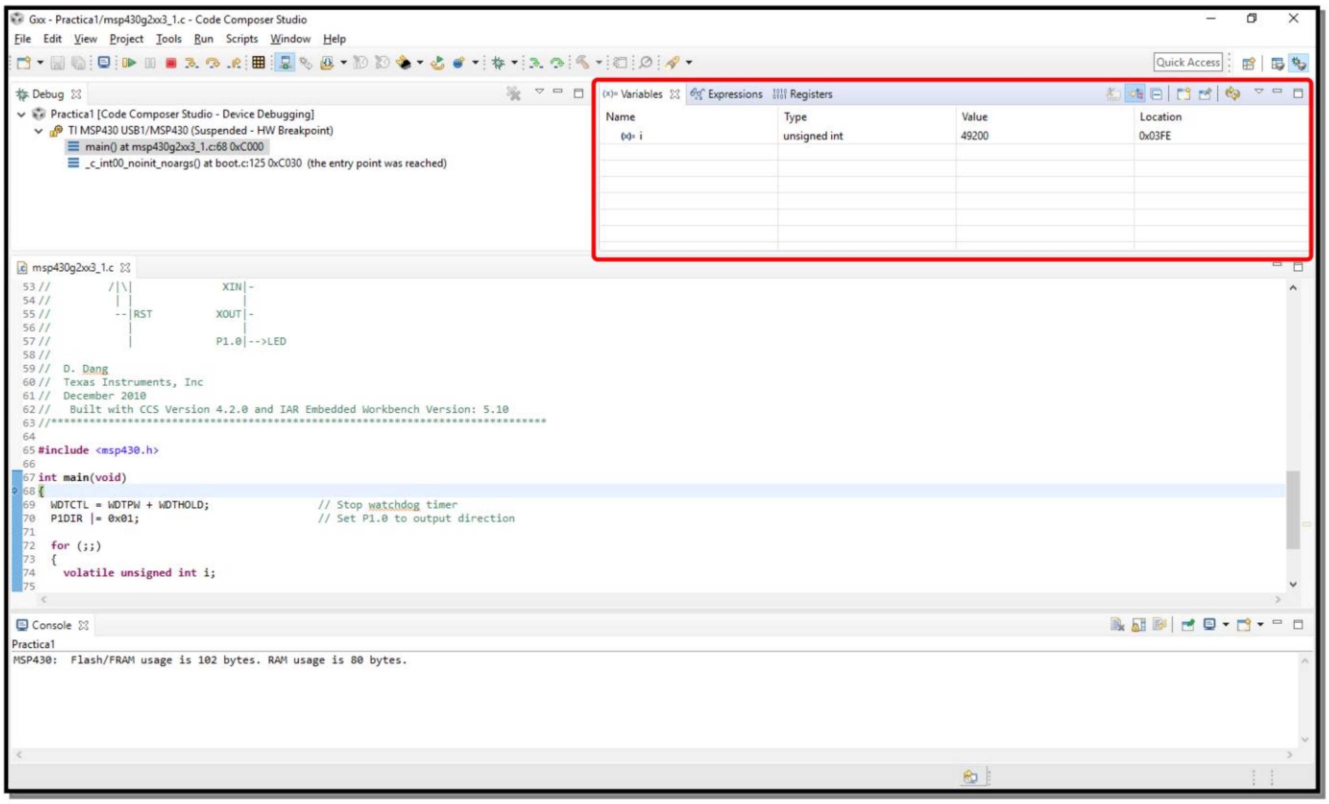
– View ÿ Variables



Name	Type	Value	Location
i	unsigned int	49200	0x03FE

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

Once the variables window is displayed, the different variables existing11 in the code are presented.


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
Variables Expressions Registers
msp430g2xx3_1.c
Console

```
Practica1 [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
    main() at msp430g2xx3_1.c:68 0xC000
    _c_int00_noinit_noargs() at boot.c:125 0xC030  (the entry point was reached)
```

Name	Type	Value	Location
i	unsigned int	49200	0x03FE

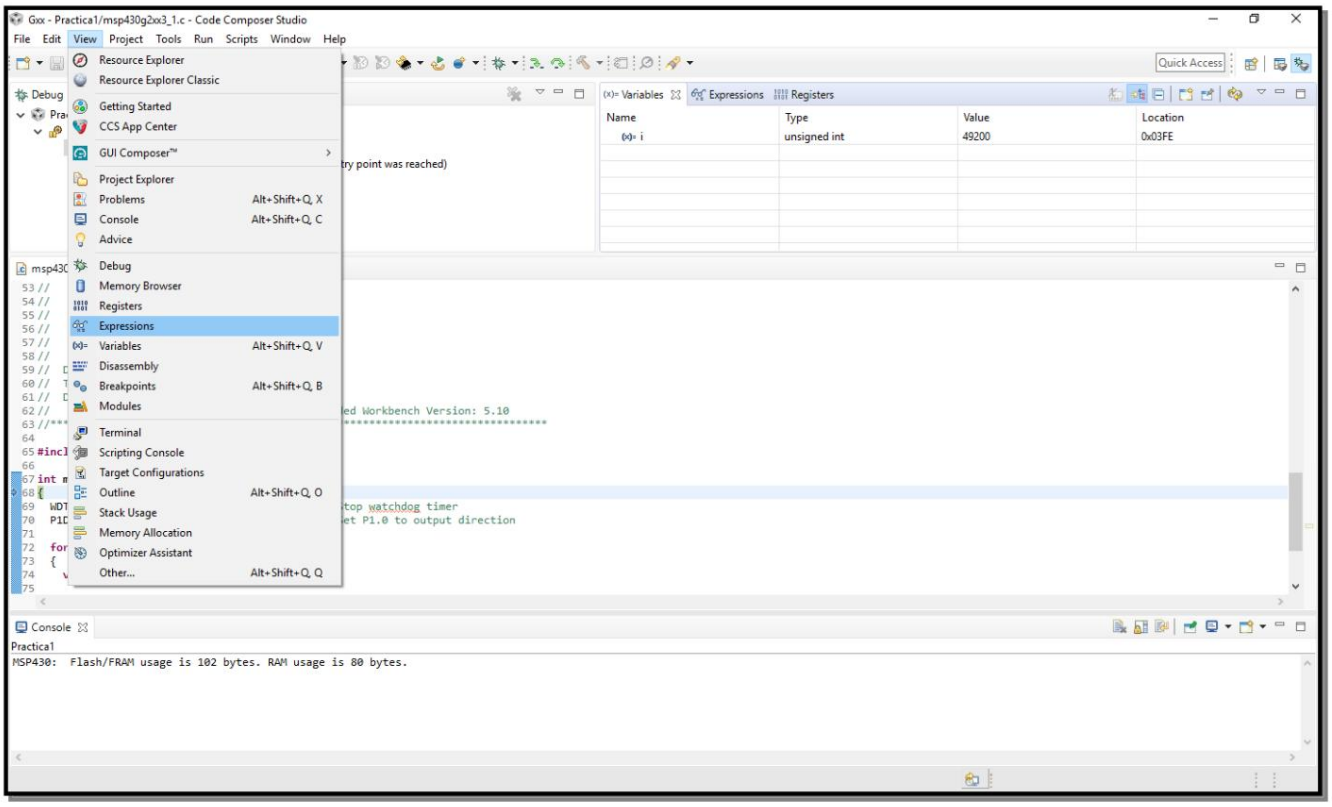
```
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|-
//            |                 |
//            |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

11 In the variables window, only the variables local to the function being executed are represented, which in this case is the function "main" that has the variable "i" defined.

If you want to inspect any other variable not shown12 in the variables window, you must use the expressions window. To show the expressions window it may be necessary to enable it with the option:

–View ÿ Expressions

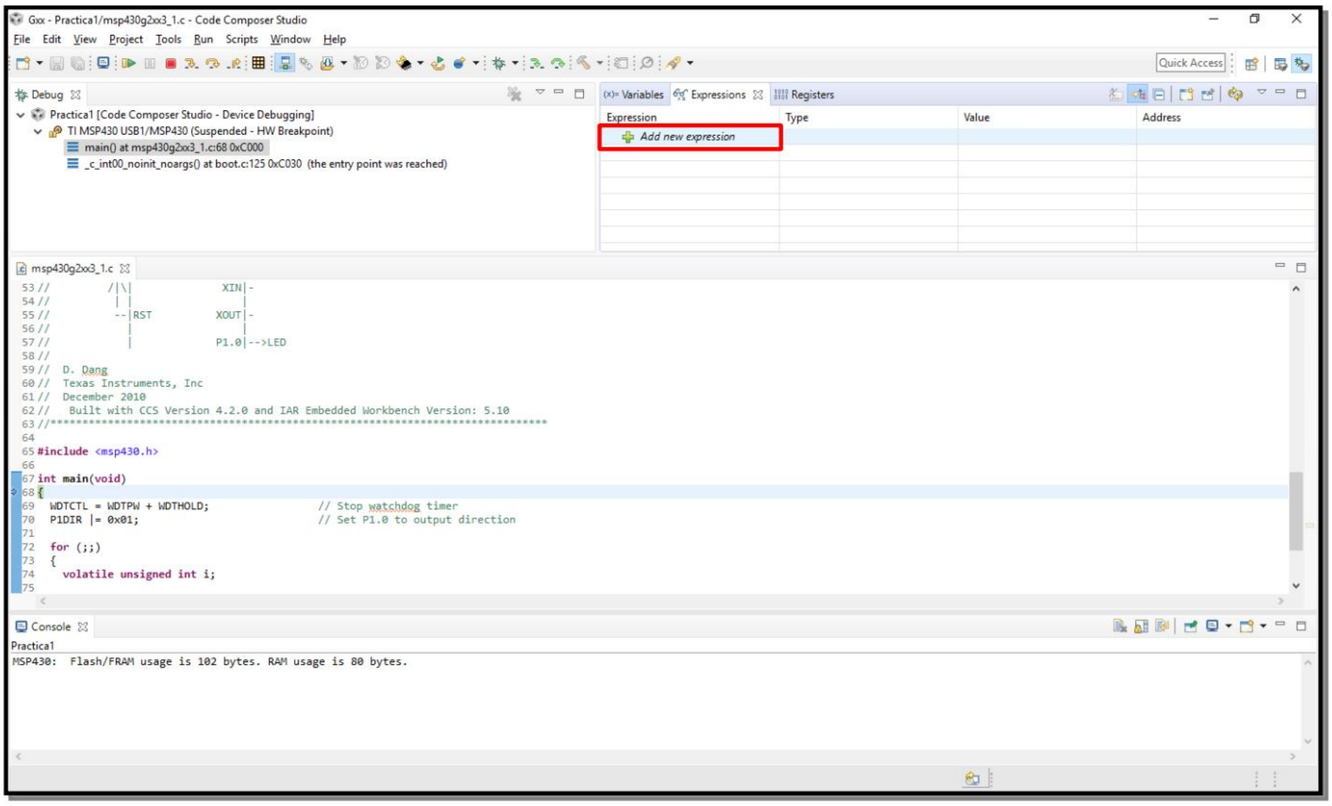


Name	Type	Value	Location
(x)= i	unsigned int	49200	0x03FE

```
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

Once the expressions window is displayed, you can enter any other variable (or expression) to be displayed with the *"Add new expression" field.*

12 Among the variables that are not shown by default in the "Variables" window are the global variables defined in the code.


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
(x)= Variables
Expressions
Registers
msp430g2xx3_1.c
Console

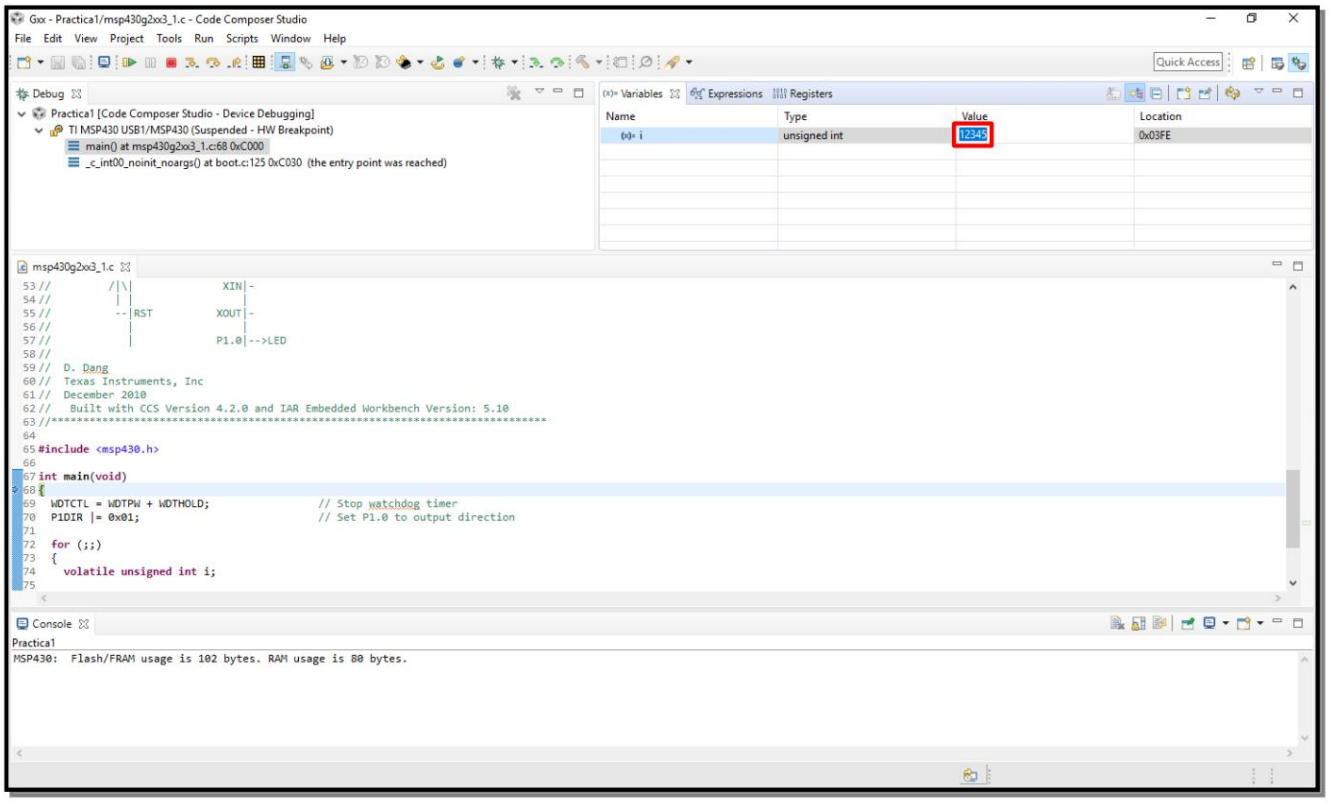
```
Practica1 [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
    main() at msp430g2xx3_1.c:68 0xC000
    _c_int00_noinit_noargs() at boot.c:125 0xC030  (the entry point was reached)
```

Expression	Type	Value	Address
Add new expression			

```
//          /|\|              XIN|-
//           | |                 |
//           --|RST          XOUT|-
//             |                 |
//             |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

Both in the variables window and in the expressions window it is possible to view and modify the value of any variable at any time. To modify it, simply select the value to be modified and enter a new one:


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
(x)= Variables
Expressions
Registers
msp430g2xx3_1.c
Console

```
Practica1 [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
    main() at msp430g2xx3_1.c:68 0xC000
    _c_int00_noinit_noargs() at boot.c:125 0xC030  (the entry point was reached)
```

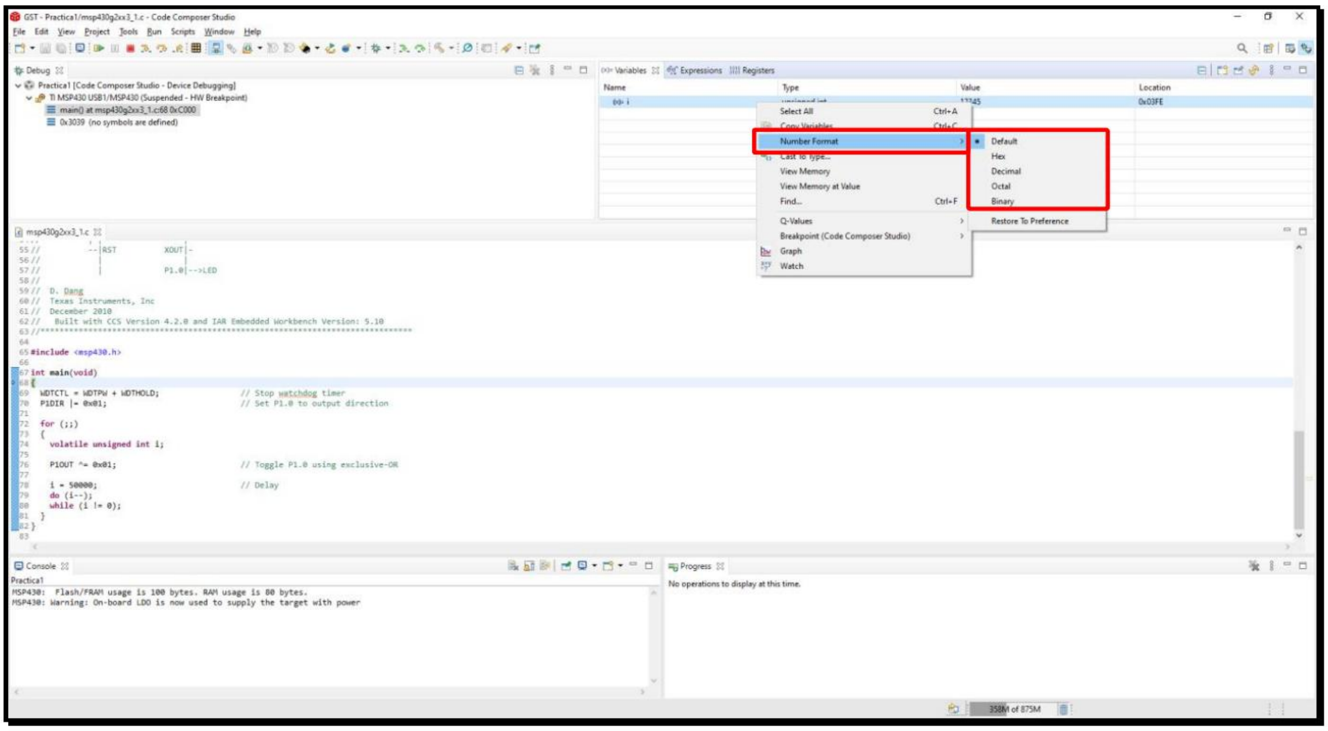
Name	Type	Value	Location
(x)= i	unsigned int	12345	0x03FE

```
//          /|\|              XIN|-
//           | |                 |
//           --|RST          XOUT|-
//             |                 |
//             |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

It is also possible to modify the format in which the data is displayed (decimal, hexadecimal, binary...), by clicking the right mouse button on the variable and selecting the desired format from the contextual menu that appears:

– Number Format ÿ Default / Hex / Decimal / Octal / Binary


GST - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Practica1 [Code Composer Studio - Device Debugging]
TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
main() at msp430g2xx3_1.c:68 0xC000
0x3039 (no symbols are defined)
Variables
Expressions
Registers
Name
Type
Value
Location
i
0x03FE
Select All Ctrl+A
Number Format
Cast To Type...
View Memory
View Memory at Value
Find... Ctrl+F
Q-Values
Breakpoint (Code Composer Studio)
Graph
Watch
Default
Hex
Decimal
Octal
Binary
Restore To Preference
msp430g2xx3_1.c
Console
Practica1
Progress
No operations to display at this time.
35M of 875M

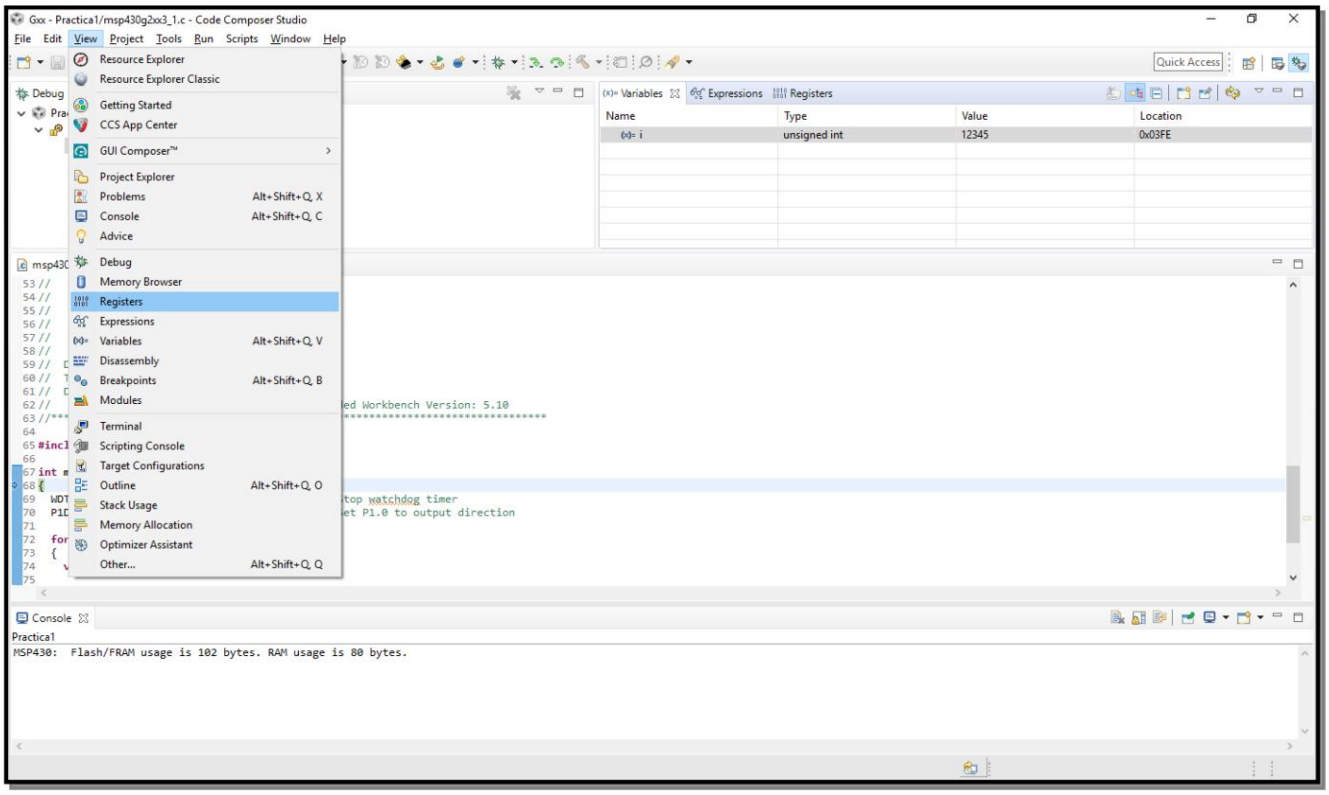
```
//             --|RST          XOUT|-
//               |                 |
//               |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-OR

    i = 50000;                              // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
MSP430:  Flash/FRAM usage is 100 bytes. RAM usage is 80 bytes.
MSP430: Warning: On-board LDO is now used to supply the target with power
```

4.2.2. records

It is also possible to view the value of all existing registers in the microcontroller, which include the peripheral configuration registers. To show the logs window it may be necessary to enable it with the option:

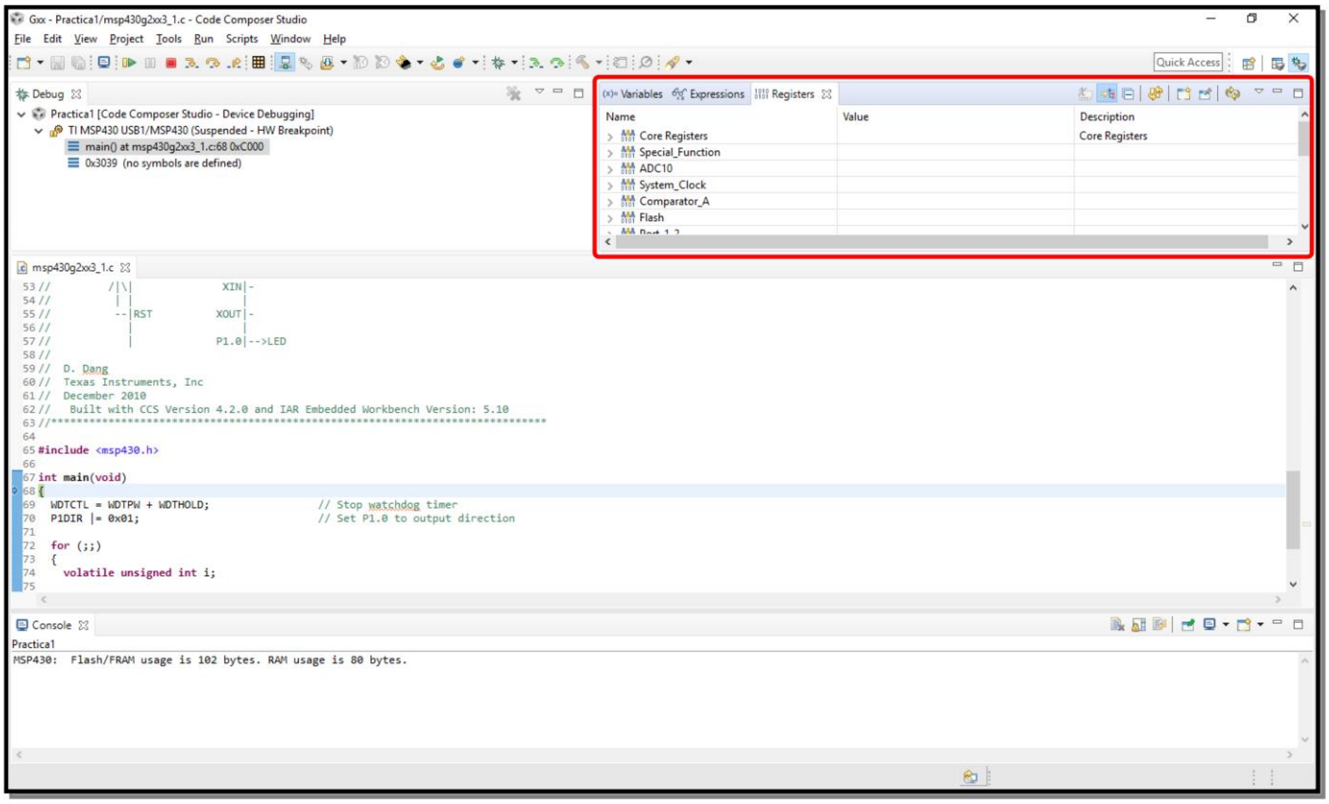
–View ÿ Registers



Name	Type	Value	Location
i	unsigned int	12345	0x03FE

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

Once the registers window is displayed, the existing registers in the microcontroller are presented.


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
Variables Expressions Registers
msp430g2xx3_1.c
Console

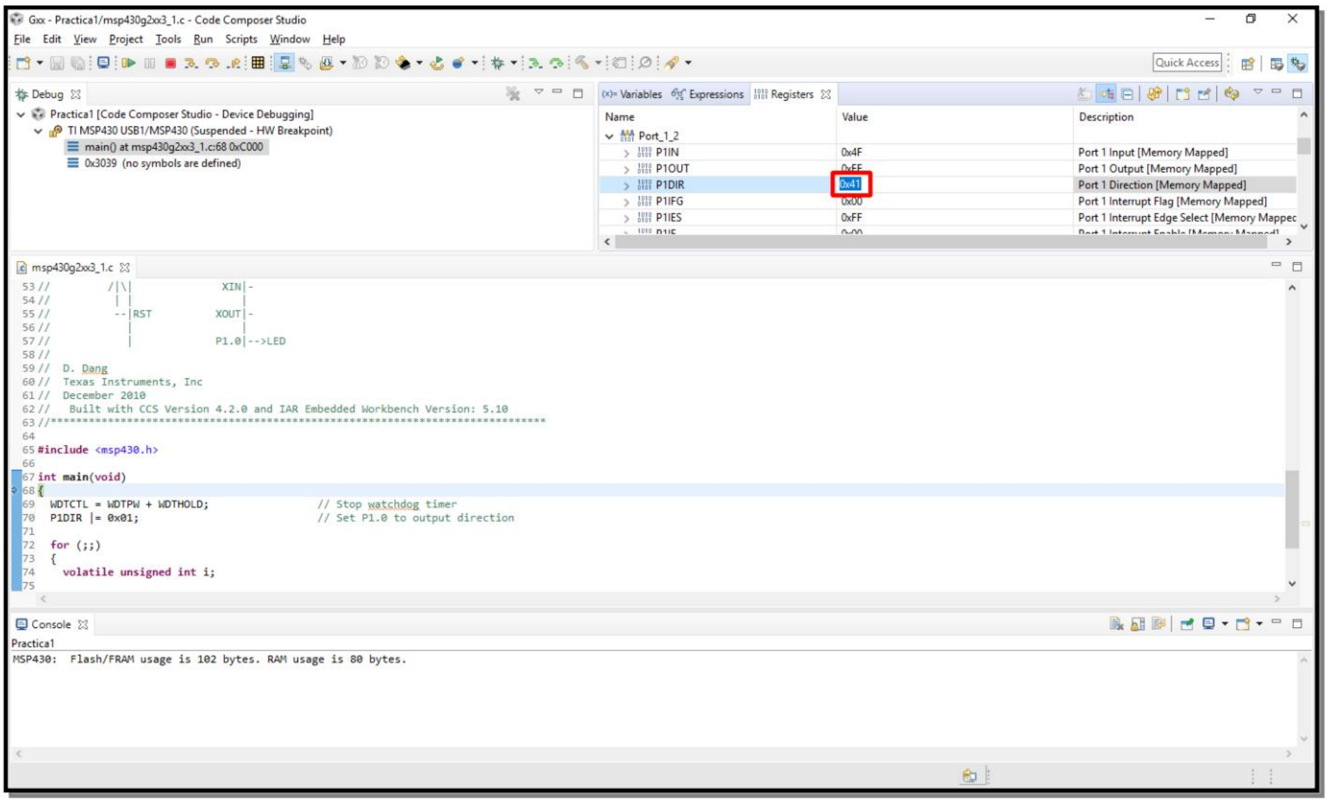
```
Practica1 [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
    main() at msp430g2xx3_1.c:68 0xC000
    0x3039  (no symbols are defined)
```

Name	Value	Description
Core Registers		Core Registers
Special_Function		
ADC10		
System_Clock		
Comparator_A		
Flash		

```
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|-
//            |                 |
//            |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

It is possible both to view and modify the value of any record at any time. To modify it, simply select the value to be modified and enter a new one:


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
Practica1 [Code Composer Studio - Device Debugging]
TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
main() at msp430g2xx3_1.c:68 0xC000
0x3039 (no symbols are defined)
Variables
Expressions
Registers
msp430g2xx3_1.c
Console

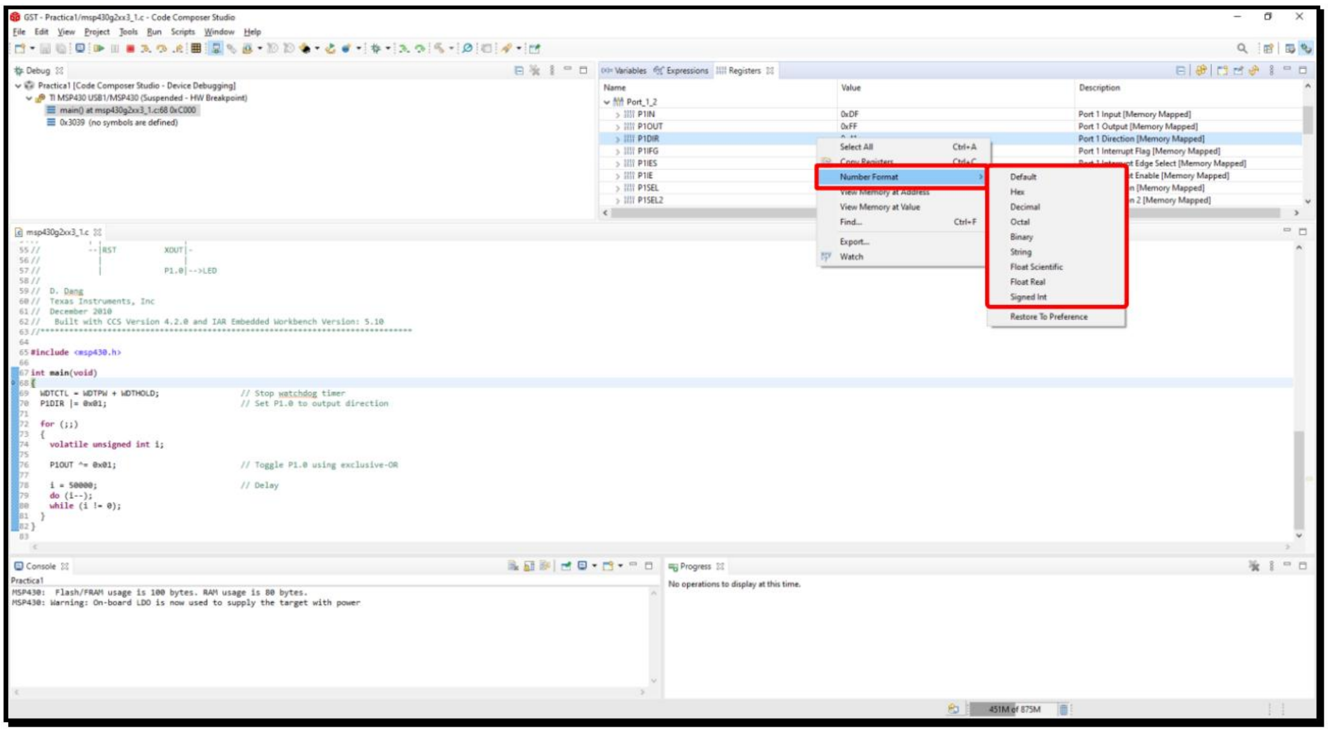
Name	Value	Description
Port_1_2		
P1IN	0x4F	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x00	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mapped]

```c
//            /|\|              XIN|-
//             | |                 |
//             --|RST          XOUT|-
//               |                 |
//               |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;
```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

It is also possible to modify the format in which the data is displayed (decimal, hexadecimal, binary...), by clicking the right mouse button on the record and selecting the desired format from the contextual menu that appears:

– Number Format ÿ Default / Hex / Decimal / Octal / Binary


GST - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Practica1 [Code Composer Studio - Device Debugging]
TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
main() at msp430g2xx3_1.c:68 0xC000
0x3039 (no symbols are defined)
Variables
Expressions
Registers
Select All Ctrl+A
Copy Registers Ctrl+C
Number Format
View Memory at Address
View Memory at Value
Find... Ctrl+F
Export...
Watch
Default
Hex
Decimal
Octal
Binary
String
Float Scientific
Float Real
Signed Int
Restore To Preference
msp430g2xx3_1.c
Console
Progress
No operations to display at this time.
451M of 875M

Name	Value	Description
Port_1_2		
P1IN	0xDF	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR		Port 1 Direction [Memory Mapped]
P1IFG		Port 1 Interrupt Flag [Memory Mapped]
P1IES		Port 1 Interrupt Edge Select [Memory Mapped]
P1IE		t Enable [Memory Mapped]
P1SEL		n [Memory Mapped]
P1SEL2		n 2 [Memory Mapped]

```c
//             --|RST          XOUT|-
//               |                 |
//               |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-OR

    i = 50000;                              // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is 100 bytes. RAM usage is 80 bytes.
MSP430: Warning: On-board LDO is now used to supply the target with power
```

4.3. Execution

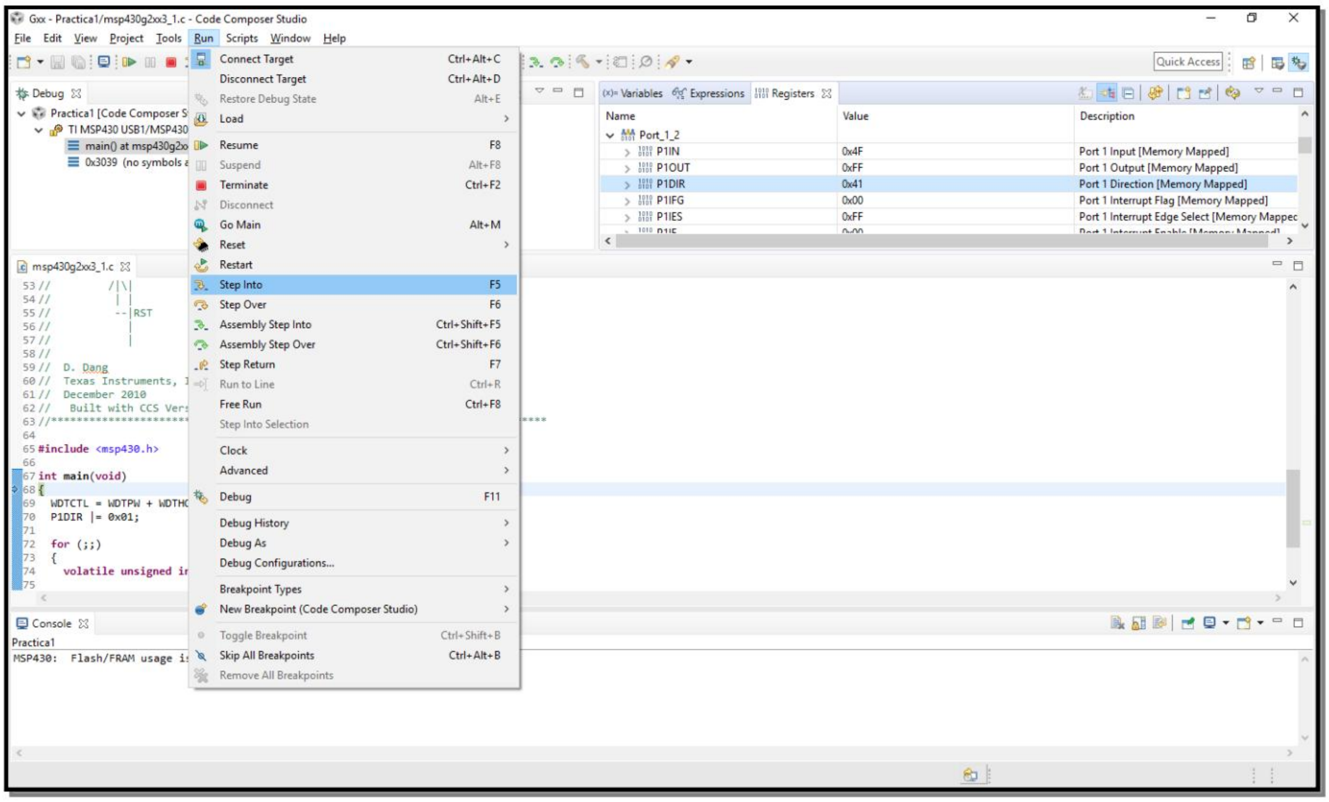
The debugger enables controlled code execution, making it easy to view and modify variables and registers during code execution. The main execution tools are described below.

4.3.1. Step by Step

The instructions can be executed one by one using the options13:

–Run ÿ Step Into

–Run ÿ Step Over



Name	Value	Description
Port_1_2		
P1IN	0x4F	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x00	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

```
//          /|\|
//           | |
//           --|RST
//             |
//             |
//
//  D. Dang
//  Texas Instruments, I
//  December 2010
//   Built with CCS Vers
//*************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHC
  P1DIR |= 0x01;

  for (;;)
  {
    volatile unsigned in
```

```
Practica1
MSP430:  Flash/FRAM usage is
```

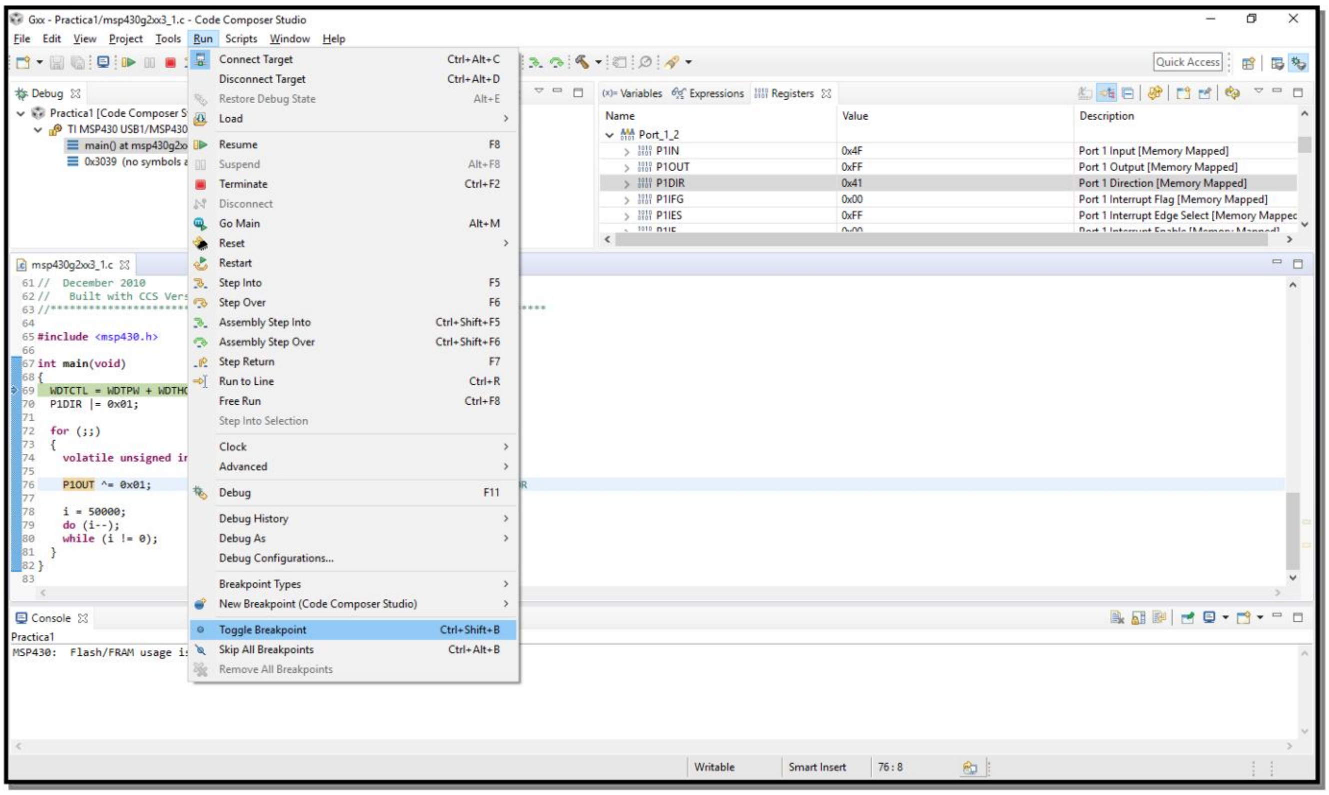
4.3.2. breaking points

Breakpoints are used to specify those statements where you want to stop code execution. To activate/ deactivate a breakpoint in a given instruction, go to the line of code that contains the instruction of interest and select 14:

–Run ÿ Toggle Breakpoint

13 The difference between the *Step Into* and *Step Over* options is in the functions, since the *Step Into* option steps the code associated with the functions while the *Step Over* option executes them completely as if they were a single instruction. In the rest of the instructions, both options are totally equivalent, so any of them can be used.

14 The *Toggle Breakpoint* action can also be performed by double- *clicking* on the left side area (shaded in grey) of the line of code where you want to activate/deactivate the breakpoint.

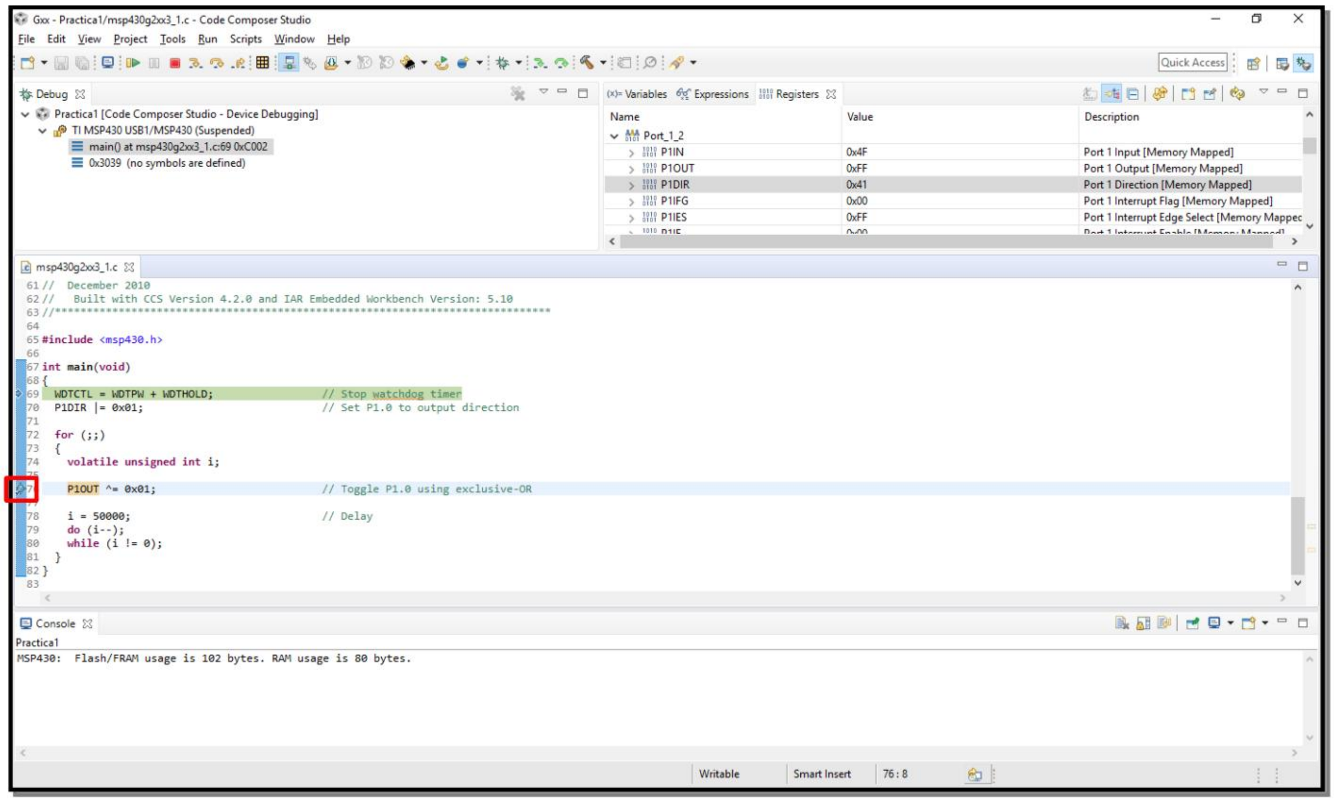

Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Connect Target Ctrl+Alt+C
Disconnect Target Ctrl+Alt+D
Restore Debug State Alt+E
Load
Resume F8
Suspend Alt+F8
Terminate Ctrl+F2
Disconnect
Go Main Alt+M
Reset
Restart
Step Into F5
Step Over F6
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
Debug F11
Debug History
Debug As
Debug Configurations...
Breakpoint Types
New Breakpoint (Code Composer Studio)
Toggle Breakpoint Ctrl+Shift+B
Skip All Breakpoints Ctrl+Alt+B
Remove All Breakpoints
Quick Access
Debug
Practica1 [Code Composer S
TI MSP430 USB1/MSP430
main() at msp430g2x
0x3039 (no symbols a
Variables Expressions Registers
msp430g2xx3_1.c
Console
Writable Smart Insert 76 : 8

Name	Value	Description
Port_1_2		
P1IN	0x4F	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x00	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

```
//  December 2010
//   Built with CCS Vers
//**********************
#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHC
  P1DIR |= 0x01;

  for (;;)
  {
    volatile unsigned ir

    P1OUT ^= 0x01;

    i = 50000;
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is
```

An active breakpoint is indicated by the symbol located on the left side (grey shaded) of the line of code where the breakpoint is located.


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
Practica1 [Code Composer Studio - Device Debugging]
TI MSP430 USB1/MSP430 (Suspended)
main() at msp430g2xx3_1.c:69 0xC002
0x3039 (no symbols are defined)
Variables Expressions Registers
msp430g2xx3_1.c
Console
Writable Smart Insert 76 : 8

Name	Value	Description
Port_1_2		
P1IN	0x4F	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x00	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

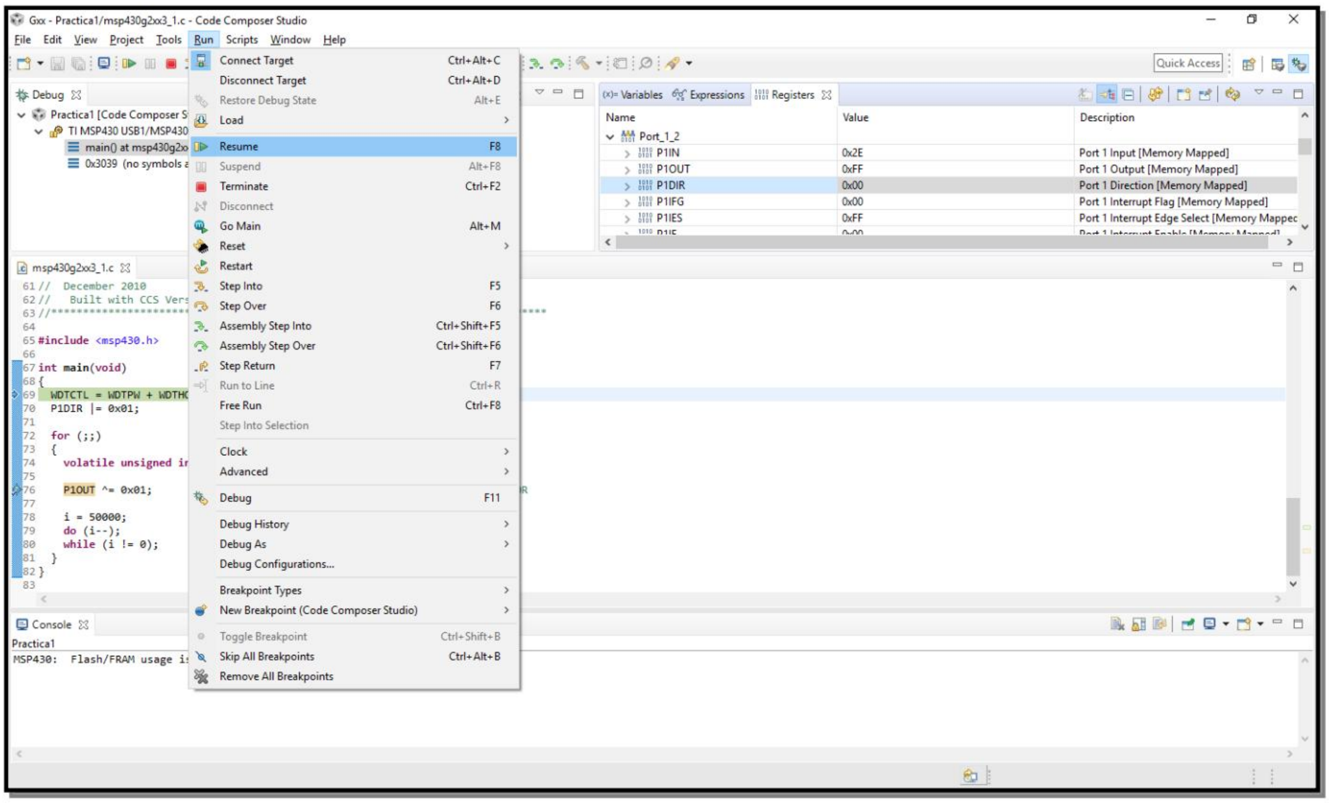
```
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-OR

    i = 50000;                              // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

4.3.3. Keep going

To run the code continuously you have to select the option:

– Run ÿ Resume

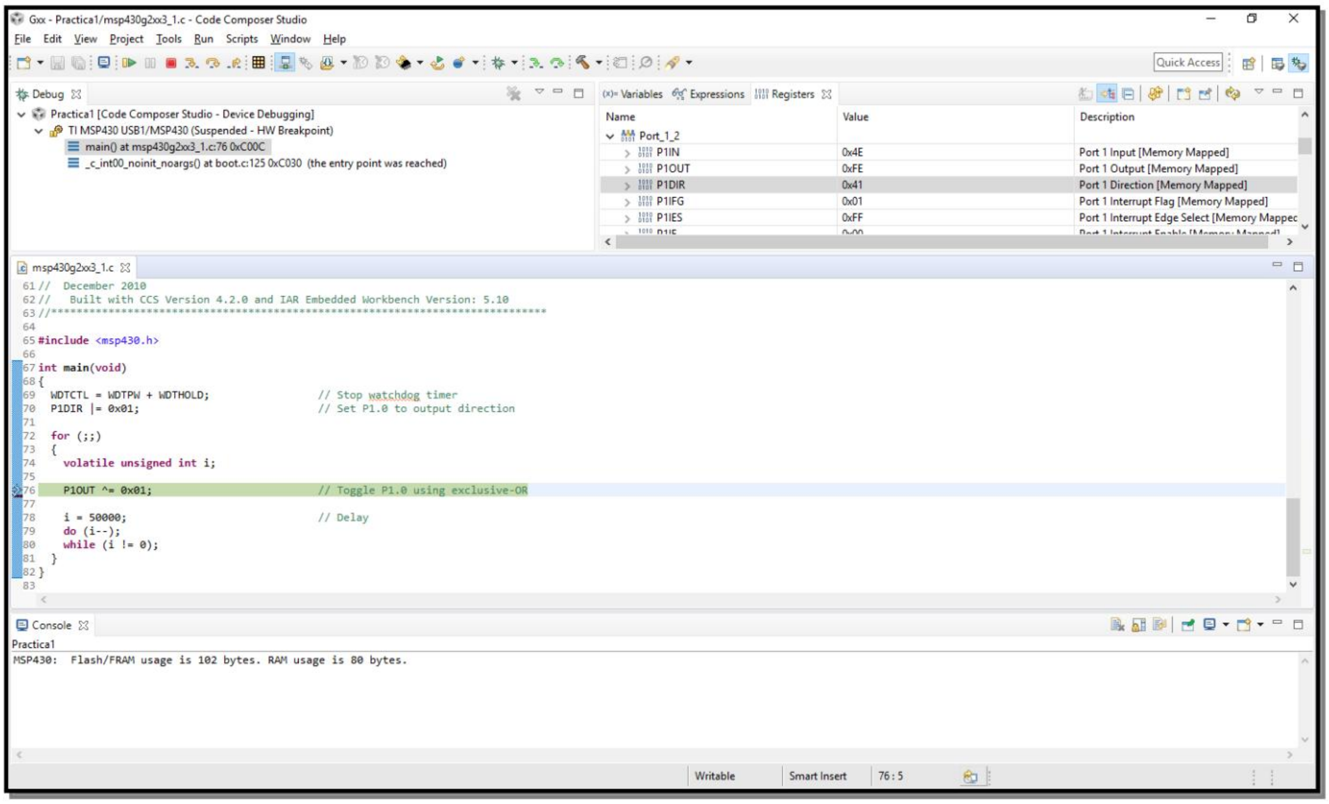


Name	Value	Description
Port_1_2		
P1IN	0x2E	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR	0x00	Port 1 Direction [Memory Mapped]
P1IFG	0x00	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

```
//   December 2010
//   Built with CCS Vers
//*********************
 
#include <msp430.h>
 
int main(void)
{
  WDTCTL = WDTPW + WDTH
  P1DIR |= 0x01;
 
  for (;;)
  {
    volatile unsigned ir
 
    P1OUT ^= 0x01;
 
    i = 50000;
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is
```

If a breakpoint has been specified in the code, the execution will stop when it is reached:



Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
(x)= Variables
Expressions
Registers
Quick Access
msp430g2xx3_1.c
Console
Writable
Smart Insert
76 : 5

```
Practica1 [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
    main() at msp430g2xx3_1.c:76 0xC00C
    _c_int00_noinit_noargs() at boot.c:125 0xC030  (the entry point was reached)
```

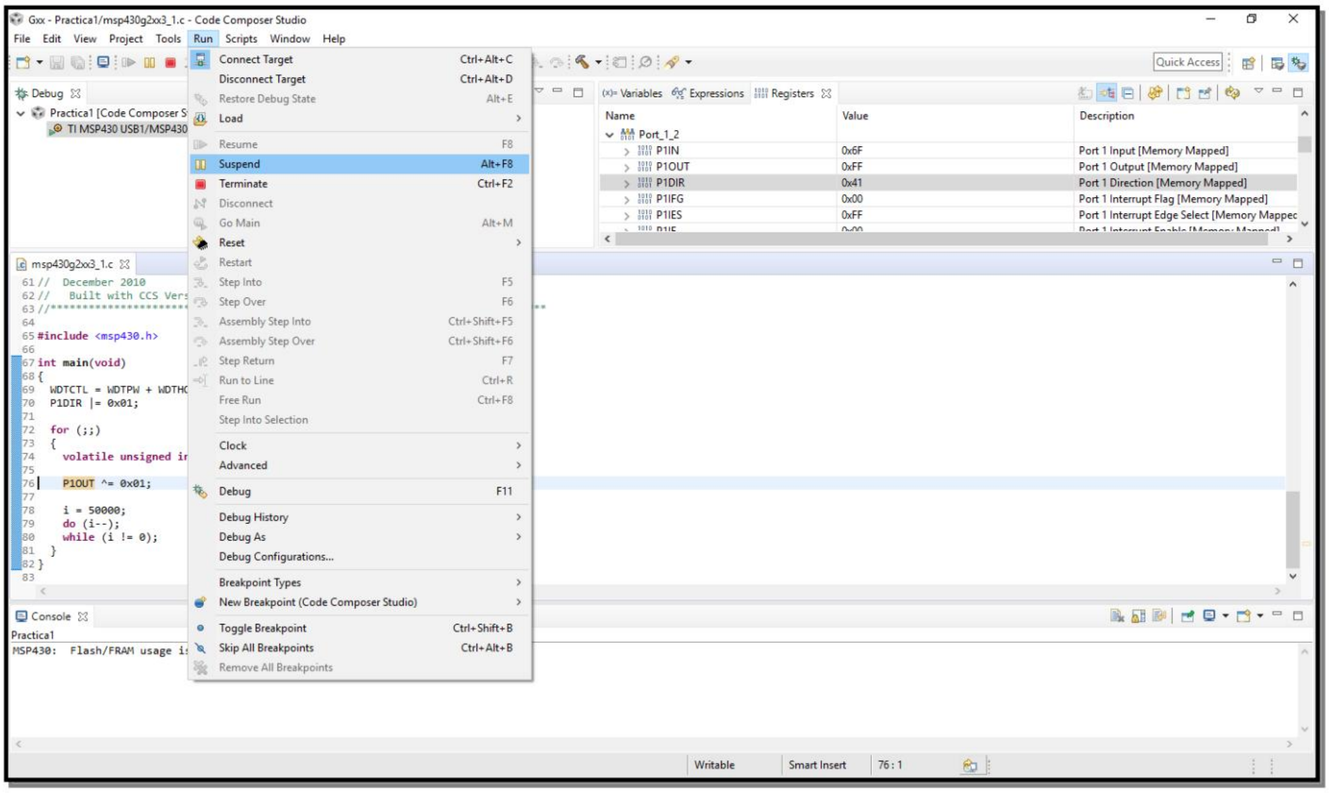
Name	Value	Description
Port_1_2		
P1IN	0x4E	Port 1 Input [Memory Mapped]
P1OUT	0xFE	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x01	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

```
//   December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************
 
#include <msp430.h>
 
int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction
 
  for (;;)
  {
    volatile unsigned int i;
 
    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-OR
 
    i = 50000;                              // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

However, if no breakpoint is specified in the code, execution will continue indefinitely until the option is given:

– Run ÿ Suspend

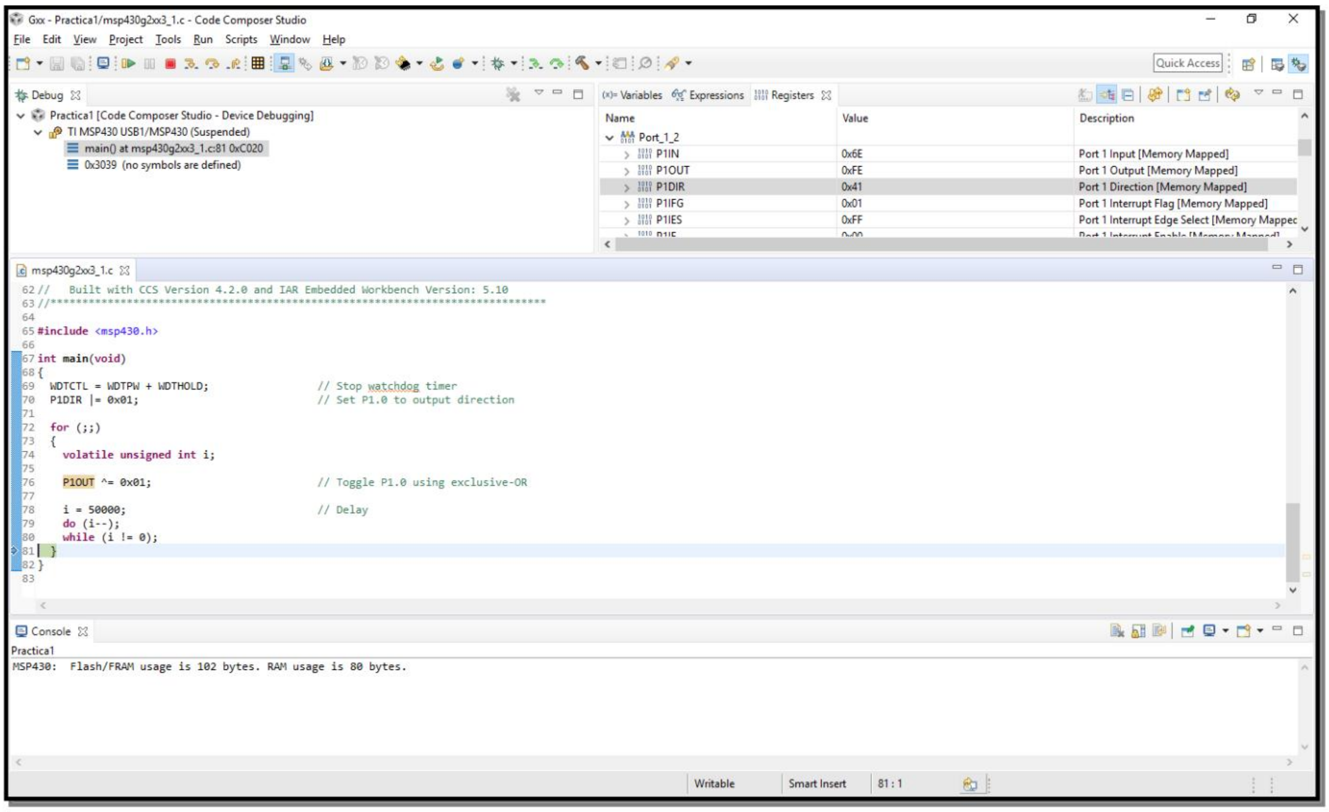

Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Connect Target Ctrl+Alt+C
Disconnect Target Ctrl+Alt+D
Restore Debug State Alt+E
Load
Resume F8
Suspend Alt+F8
Terminate Ctrl+F2
Disconnect
Go Main Alt+M
Reset
Restart
Step Into F5
Step Over F6
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
Debug F11
Debug History
Debug As
Debug Configurations...
Breakpoint Types
New Breakpoint (Code Composer Studio)
Toggle Breakpoint Ctrl+Shift+B
Skip All Breakpoints Ctrl+Alt+B
Remove All Breakpoints
Debug
Practica1 [Code Composer S
TI MSP430 USB1/MSP430
Variables Expressions Registers
Console
Quick Access
Writable Smart Insert 76 : 1

Name	Value	Description
Port_1_2		
P1IN	0x6F	Port 1 Input [Memory Mapped]
P1OUT	0xFF	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x00	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

```
//   December 2010
//   Built with CCS Vers
//**********************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHC
  P1DIR |= 0x01;

  for (;;)
  {
    volatile unsigned in

    P1OUT ^= 0x01;

    i = 50000;
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is
```

moment in which the execution of the code is stopped, showing the instruction that was going to be executed next:


Gxx - Practica1/msp430g2xx3_1.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
Variables Expressions Registers
Console
Writable Smart Insert 81 : 1

Practica1 [Code Composer Studio - Device Debugging]
TI MSP430 USB1/MSP430 (Suspended)
main() at msp430g2xx3_1.c:81 0xC020
0x3039 (no symbols are defined)

Name	Value	Description
Port_1_2		
P1IN	0x6E	Port 1 Input [Memory Mapped]
P1OUT	0xFE	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x01	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

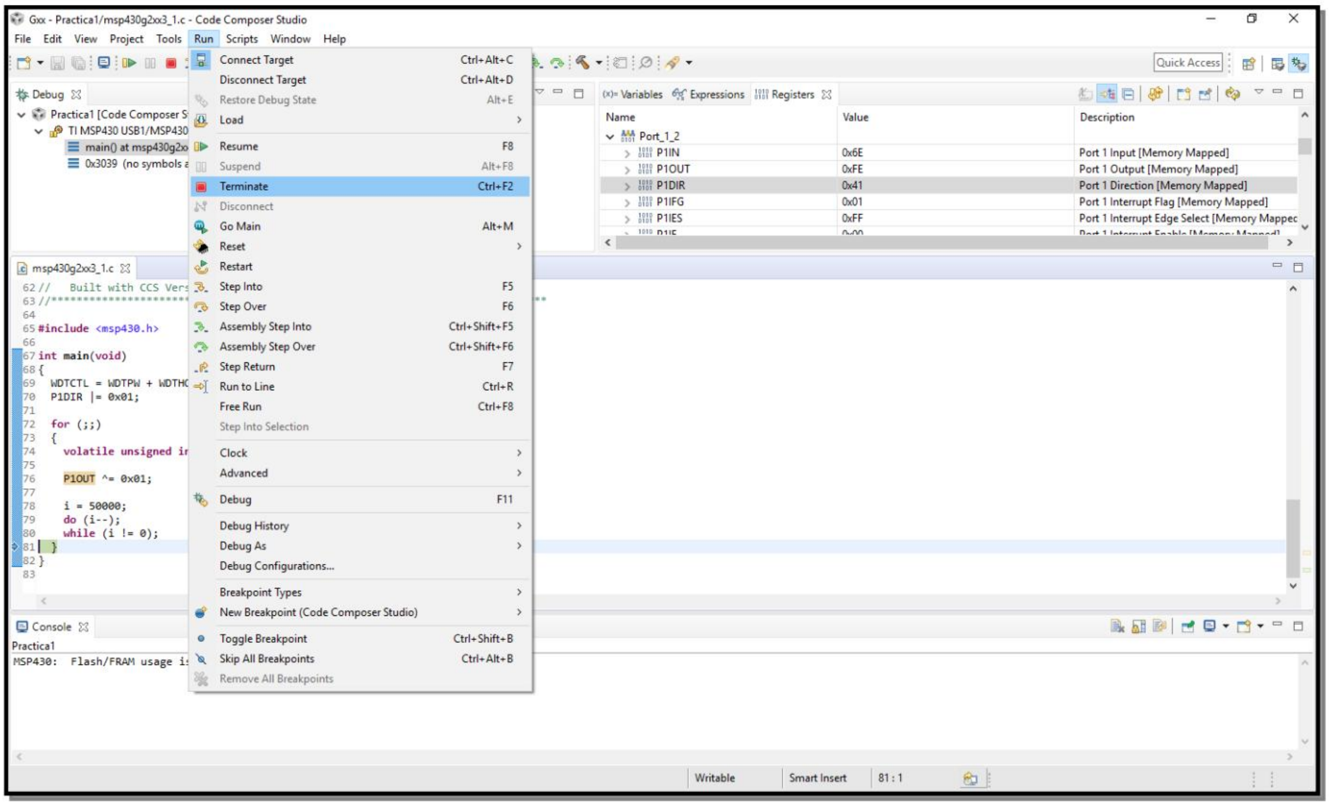
```
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-OR

    i = 50000;                              // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

4.4. End of debuggingión

To end the debugging process, select the option:

–Run ÿ Terminate

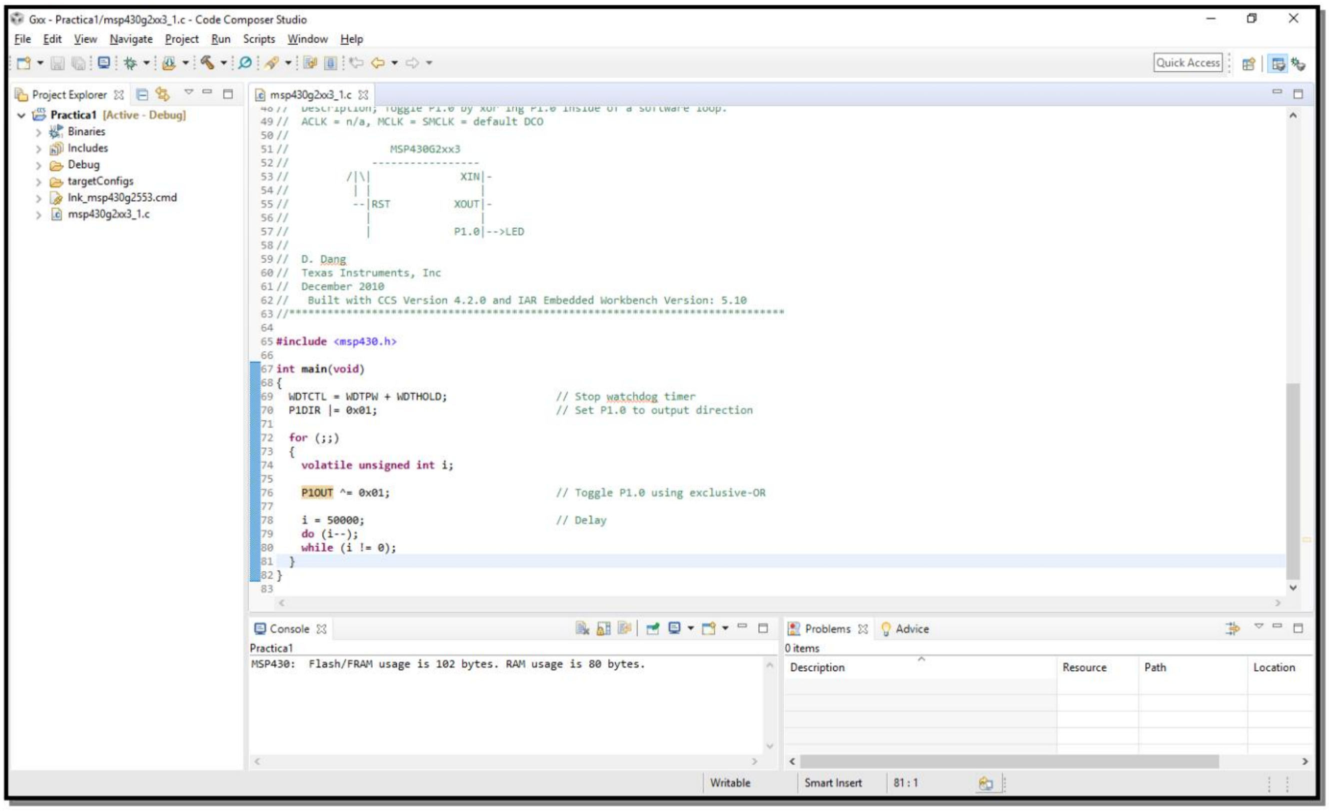


Name	Value	Description
Port_1_2		
P1IN	0x6E	Port 1 Input [Memory Mapped]
P1OUT	0xFE	Port 1 Output [Memory Mapped]
P1DIR	0x41	Port 1 Direction [Memory Mapped]
P1IFG	0x01	Port 1 Interrupt Flag [Memory Mapped]
P1IES	0xFF	Port 1 Interrupt Edge Select [Memory Mappec

```
//   Built with CCS Vers
//**********************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHC
  P1DIR |= 0x01;

  for (;;)
  {
    volatile unsigned ir

    P1OUT ^= 0x01;

    i = 50000;
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage i
```

Once the debugging process is finished, we return to the editor:



```
//  Description; Toggle P1.0 by xor'ing P1.0 inside of a software loop.
//  ACLK = n/a, MCLK = SMCLK = default DCO
//
//                MSP430G2xx3
//             -----------------
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|-
//            |                 |
//            |             P1.0|-->LED
//
//  D. Dang
//  Texas Instruments, Inc
//  December 2010
//   Built with CCS Version 4.2.0 and IAR Embedded Workbench Version: 5.10
//******************************************************************************

#include <msp430.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= 0x01;                            // Set P1.0 to output direction

  for (;;)
  {
    volatile unsigned int i;

    P1OUT ^= 0x01;                          // Toggle P1.0 using exclusive-OR

    i = 50000;                              // Delay
    do (i--);
    while (i != 0);
  }
}
```

```
Practica1
MSP430:  Flash/FRAM usage is 102 bytes. RAM usage is 80 bytes.
```

5. EXERCISE

Once the use of the *Code Composer Studio* development environment used in the course has been described, the following exercise is proposed to practice the concepts described in this tutorial15:

Determine the minimum values, both in decimal and hexadecimal, taken by the variables "val" and "reg" and the register "P1IES" (which belongs to the peripheral "Port 1") throughout the code, indicating the lines where they reach said values.

```
1. // edu@uma.es - 02/18/22 (v2) 2.

3. // Debugging: Determine the minimum values, both in decimal and 4. // hexadecimal, taken by the variables "val" and "reg"
and the register "P1IES" (which 5. // belongs to the peripheral "Port 1") throughout the code, indicating the lines 6. // where these values
reach 7. 8. #include "msp430g2553.h" 9. 10. unsigned char reg = 0xFF; 11. 12. void main(void) { 13. volatile unsigned int val = 0xFFFF;
14. int temp = 0; 15. unsigned int cont; 16. 17. 18. 19. 20.

21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32.

33. 34. 35. 36. 37. } // void main( void)




            WDTCTL = (WDTPW|WDTHOLD);

            for (cont = 0; count < 1000; count++) { temp += 12345; } //
                  for (count = 0; count <
            1000; count++)

            val = (temp^0x7FFF); reg =
            (unsigned char) (val^0x69);
            P1IES = (reg^0x5A);

            for (cont = 1000; count > 0; count--) { temp -= 54321; } // for
                  (count = 1000; count
            > 0; count--)

            val = (temp^0x7FFF); reg =
            (unsigned char) (val^0xE9);
            P1IES = (reg^0xDA);

            while (1);
```

15 The code of this exercise is available in the file "Depuracion.c" of the Virtual Campus of the subject.