

ENTORN DE TREBALL: CODE COMPOSER STUDIO

El programa Code Composer Studio és un Entorn de Desenvolupament Integrat (IDE en anglès) que permet el desenvolupament de projectes amb microcontroladors de Texas Instruments. Es pot programar tant en assemblador "ASM" com en "C". A més, segons quines versions, el Code Composer Studio inclou un sistema operatiu en temps real anomenat TI-RTOS, inclou també simuladors y llibreries dels microcontroladors.



FIGURA 1. Esquerra: icona del escriptori del CCS. Dreta: Imatge de càrrega del entorn

INSTAL·LACIÓ

IMPORTANT: CAL INSTAL·LAR LA VERSIÓ 10.4, VERSIONS MÉS NOVES NO FUNCIONARAN AMB EL NOSTRE HARDWARE

En el campus trobareu el link directe de descarrega per Windows de la versió 10.4. En cas d'altres sistemes operatius, podeu trobar els links (recordeu, ha de ser la 10.4 igualment) a https://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html#code-composer-studio-version-10-downloads

1. És recomana fer servir el "Single file installer" en lloc del on-demand, ja que aquest últim històricament sol tenir problemes.
2. Un cop descarregat, també és recomanable fer clic dret sobre el instal·lador i seleccionar "Executar com a administrador". En cas contrari en certs equips no funcionen correctament els divers necessaris per comunicar-se amb la placa de desenvolupament.
3. Durant la instal·lació se'ns preguntarà quins components volem instal·lar. Cal seleccionar com a mínim el SimpleLink MPS432 (Figura 2).

Durant tot el curs feu servir la mateix versió, no és recomanable actualitzar la versió instal·lada si ha funcionat correctament.

Un cop finalitzada la instal·lació, procedirem a executar per primer cop el Code Composer Studio (CCS).

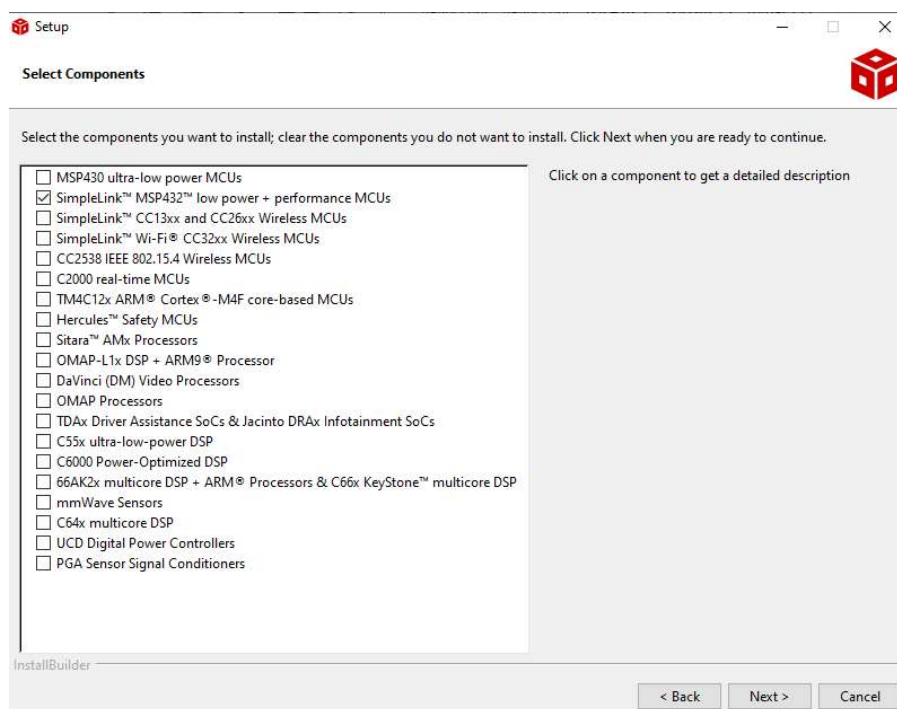


FIGURA 2. Selecció dels components a instal·lar

SELECCIÓ DEL WORKSPACE

crear un directori pel nostre "Workspace". Dintre d'aquest, després s'aniran creant automàticament carpetes per a cada un dels projectes que farem durant el curs. Iniciem l'execució de l'entorn de desenvolupament.

El programa ens demanarà el directori del nostre "Workspace". Per modificar-lo premeu l'opció "Browse". I un cop modificat, feu servir el mateix durant tota l'assignatura. En el cas dels ordinadors dels laboratoris, seleccioneu una ubicació del disc D:, tal com pot ser D:\CCS_workspace. No utilitzeu el disc C: donat que la carpeta serà eliminada al reiniciar-se l'equip.

En els vostres equips personals, eviteu fer servir rutes massa llargues, amb espais o caràcters amb accents o no d'ús comú anglès (e.g. ñ, ç). És recomana, similar al cas anterior, que en l'equip personal utilitzeu també C:\CCS_workspace

Si és el primer cop que executem el programa després de la instal·lació, ens apareixerà una pantalla de presentació que ens permet executar, si estem en línia, *tutorials* i exemples.

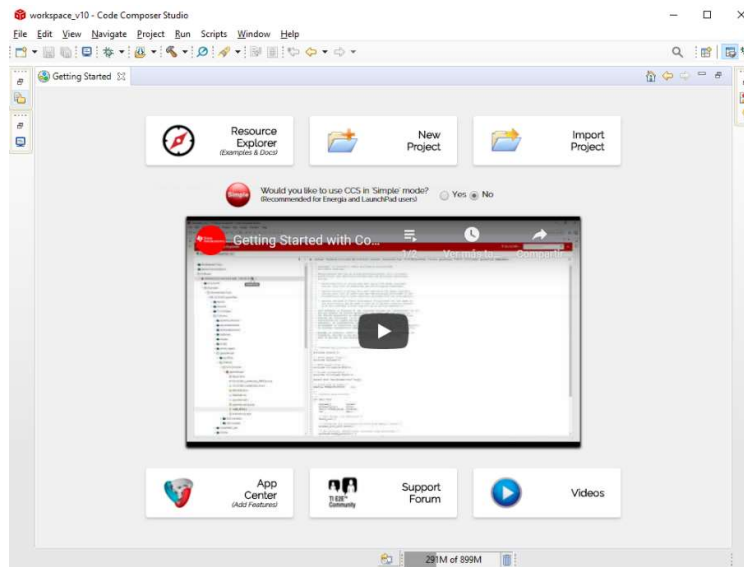


FIGURA 3. Vista de presentació inicial del CCS

CREACIÓ D'UN NOU PROJECTE

Per programar el microcontrolador el primer que farem és crear un nou projecte, seleccionant *Project -> New CCS Project...*

Els camps importants aquí són (Figura 4):

1. **Target: MSP432 Family i MSP432P401R.** Amb aquests ajustaments del projecte, triem bàsicament el dispositiu exacte amb el que volem treballar. Seleccionem el microcontrolador MSP432P401R, que és el que hi ha a la placa i que serà el que programarem per controlar el nostre robot.
2. **Project Name:** On posarem el nom que vulguem donar-li al nostre projecte.
3. Verificar on volem desar el projecte a "Location". Si marquem a "Use default location" ens crearà automàticament una carpeta pel nou projecte al nostre Workspace.

No hem de modificar res més a aquesta pantalla. En qualsevol cas, després podem canviar qualsevol d'aquests paràmetres des de *Project -> Properties*

Per últim, seleccionem *Finish*.

A continuació, entrarem a la pantalla on editarem els programes i arxius que formaran el projecte.

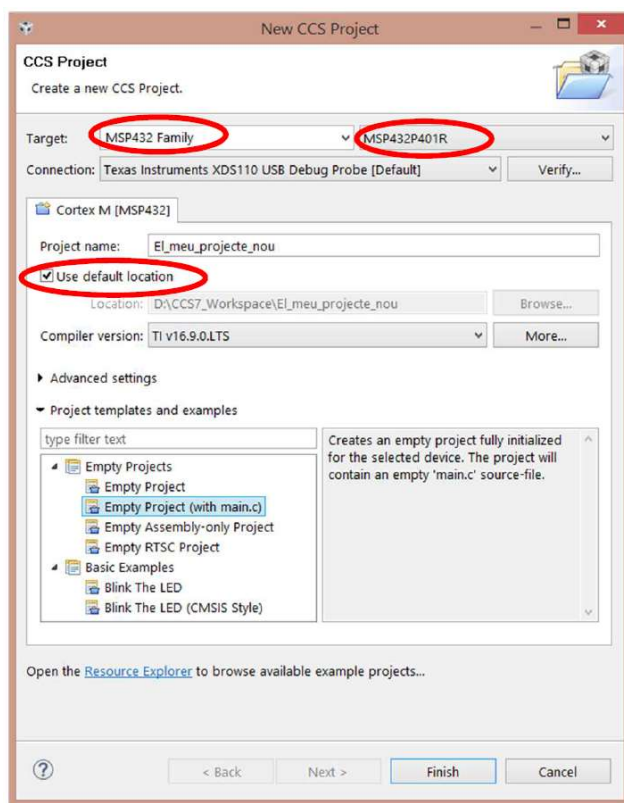


FIGURA 4. Creació d'un nou projecte del CCS

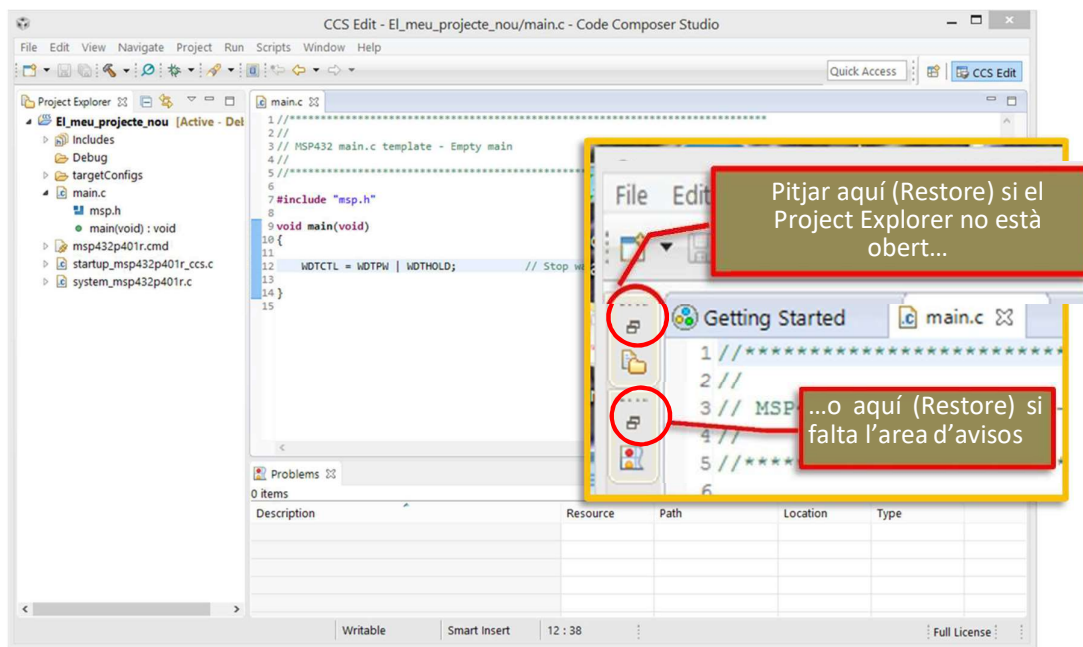


FIGURA 5. Creació d'un nou projecte del CCS

Aquesta pantalla té 3 àrees (a més de les barres de menú i eines, Figura 5):

- A l'esquerra tenim el **Project Explorer**, on es mostra l'estructura de fitxers que formen

part del projecte. Podem desplegar el nostre projecte per veure els diferents fitxers que el componen. També podem desplegar un fitxer per visualitzar les variables, funcions, etc... que conté, i fent doble clic sobre qualsevol d'aquests elements, podem accedir ràpidament al codi de la seva declaració. Si tenim més d'un projecte al workspace, **el projecte actiu estarà marcat amb negreta**.

- Al centre està l'àrea d'Edició, mostra el contingut dels fitxers que tinguem oberts per editar.
- A baix està l'àrea d'avisos, que mostra tots els problemes detectats durant la compilació (errors, warnings...).

Si falta qualsevol d'aquestes àrees, i tampoc apareix el seu respectiu botó de *Restore*, haurem d'anar al menú *Window -> Show View* i seleccionar la que volem tornar a veure.

A l'exemple anterior veiem que el projecte està carregat. A més dels *"includes"* i arxius de configuració necessaris per al microcontrolador que hem triat per treballar i que s'han afegit automàticament, hi tenim 3 arxius de programa que s'han creat, també automàticament (*.c). Tots aquests arxius són imprescindibles. Només hauríem d'editar el *main.c* o qualsevol que hàgim afegit nosaltres, els altres de moment són "NO TOCAR".

Les instruccions creades automàticament dins l'arxiu *main.c* són les mínimes obligatòries que hauria de contenir sempre l'arxiu principal de qualsevol dels nostres projectes per desenvolupar aplicacions amb el microcontrolador seleccionat.

CREACIÓ D'UN ARXIU DE CODI ".C" O ".H"

Per crear un arxiu de programa ".c" nou on poder picar codi de programa hem de fer: *File->New -> Source File*

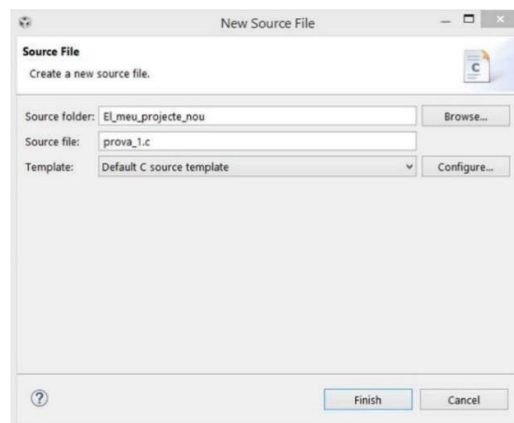


FIGURA 6. Creació d'un nou fitxer font

A **Source File** hem de ficar el nom de l'arxiu, no ens hem d'oblidar de posar l'extensió, en aquest cas ".c".

A **Template**, hem de seleccionar el corresponent al llenguatge C (no C++).

Donem a **Finish** i ja tindrem el nou fitxer a l'estructura del projecte a l'àrea esquerra.



Si volem crear un arxiu de capçaleres “.h” el procediment és el mateix, però en lloc de *New Source File*, hem de triar **New Header File**, tenint en compte sempre que hem de posar l’extensió “.h” quan posem el nom del fitxer.

Per **compilar** un projecte hem de fer: *Project -> Build Project*.

Un cop finalitzada la compilació, els possibles problemes ens apareixeran a l’àrea “*Problems*” sota la d’edició. Haurem de parar sempre molta atenció a aquesta àrea que ens pot mostrar 2 tipus de problemes:

- “**Errors**”: aquests són crítics i només que n’hi hagi un, no permetrà l’execució del programa. De vegades poden sortir molts errors, no hem de desesperar ja que en moltes ocasions quan solucionem un error desapareixen molts dels altres en fer un nou **Build**. És important **mirar la consola de compilació** per entendre millor el context del error.
- “**Warnings**”: no són crítics i el programa pot executar-se, però pot ser que no faci exactament el que volem perquè hi ha algun desajust de tipus. **En el vostre nivell actual, heu de considerar tots els warnings com a errors i solucionar-los.**

Si fem un Build del nostre projecte actual, veiem que no ens surt cap error ni Warning... de moment! El que passa per ara és que no tenim cap programa fet.

Fem un petit programa de prova per veure com funcionen aquests passos. El nostre programa no farà res per ara, per tant només posarem els *includes* mínims i deixarem la rutina principal, la **main()**, sense cap instrucció.

Sempre que creem un nou document “.c” hem de tenir en compte que haurem d’afegir a l’inici els **includes** del nostre processador (**#include "msp.h"**), de les funcions estàndard d’entrada/sortida (**#include <stdio.h>**), i dels tipus estàndards d’enters (**#include <stdint.h>**).

A més, la primera instrucció de la funció main() sempre haurà de ser:

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

(la raó de ser d’aquesta instrucció s’explicarà més endavant en una pràctica posterior).

Això si, sempre que fem programes a baix nivell, hem de comentar tot molt bé, des del que fa cada funció, els paràmetres d’entrada i sortida que fa servir, fins a pràcticament cada instrucció.

Per tant, és un hàbit **molt recomanable anar afegint els comentaris “sobre la marxa”** (en comptes d’esperar a que estigui tot el codi per escriure els comentaris), és a dir, cada cop que es pica una línia de codi, afegir immediatament un comentari **pertinent i útil**.

```
/*  
 *   PROGRAMA DE PROVA.  
 */  
  
#include <msp.h>  
#include <stdio.h>  
#include <stdint.h>  
  
#include "prova.h"
```

```
void main()
{
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
    while(1);
}
```

Creem un fitxer **Header** anomenat *prova.h* que no tingui res i provem a compilar-ho per veure que no hi ha errors. Si hi ha errors ens ho mirem bé perquè serà el programa més senzill que farem a l'assignatura...

Com hem comentat abans, si volem canviar la configuració del nostre projecte ho podem fer en qualsevol moment a: *Project -> Properties*
Ens apareixerà la següent pantalla:

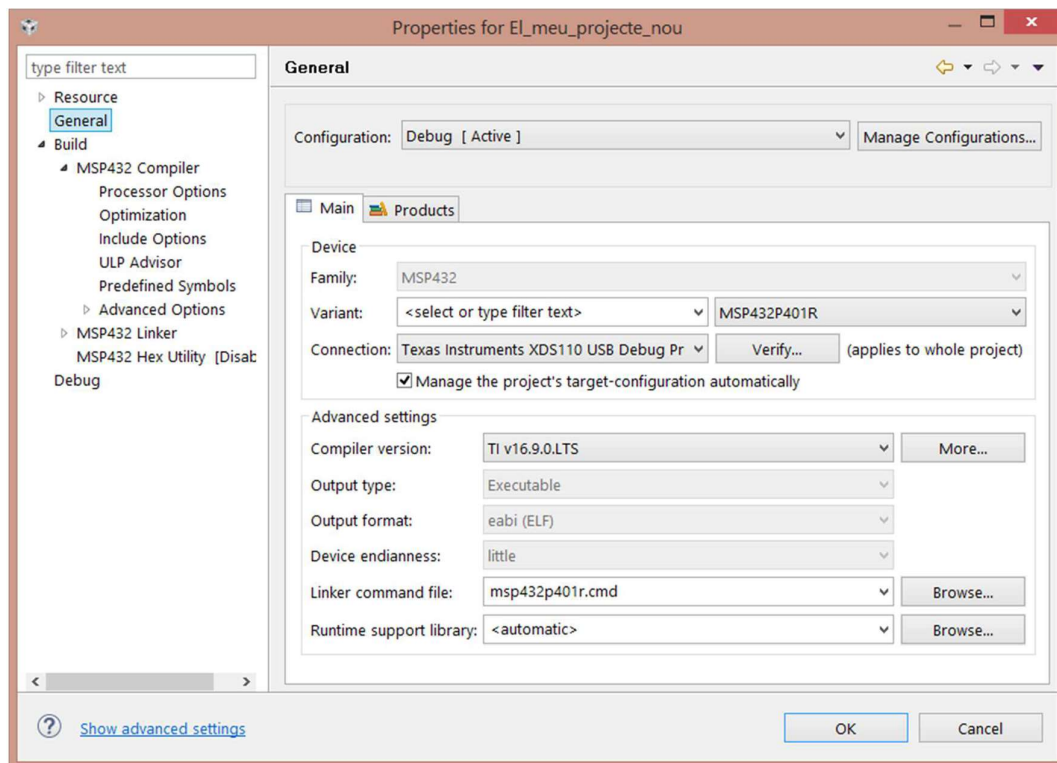


FIGURA 7. Propietats del projecte

Si triem **General**, podem modificar el microcontrolador que fem servir. Amb les altres opcions es poden modificar moltes opcions del compilador.

Generar els fitxers que formaran el nostre projecte és el primer pas a la nostra tasca de programar el microcontrolador. Fins aquí el programa està al PC, ara hem de carregar el programa al microcontrolador i executar-lo per veure si fa el que volem que faci. Això és el que anomenarem **“Debug del projecte”**.

BARRA D'EINES


Disposem de diversos accessos ràpids als comandaments més usuals mitjançant la barra d'eines:



Els més útils per ordre: ... **Desar**, ... **Build** (el martell), ..., **Debug** (l'insecte), **Search** (la llanterna), ... etc.

DEPURAR (“DEBUGAR”) UN PROGRAMA

Per carregar el programa al microcontrolador i després poder fer la depuració del programa, hem de connectar la placa on es troba el microcontrolador (*Launchpad MSP-EXP432P401R*) al PC mitjançant el seu cable USB. Al *Code Composer Studio* (CCS) farem una d'aquestes accions per començar la càrrega del programa i la depuració:

- Mitjançant el menú **Run -> Debug** (o F11).
- O bé mitjançant l'accés ràpid en forma d'insecte (“bug”)  de la Barra d'Eines.

Si encara no hem desat el programa al PC ni l'hem compilat, el CCS ens demanarà de fer-ho. (Si ens surt un missatge demanant que féssim una actualització de firmware, li diem “Update”, i quan acabi aquest procés, es reprendrà la compilació).

Després, si no hi ha cap error més, entrarem automàticament a la pantalla de depuració o *debug*:

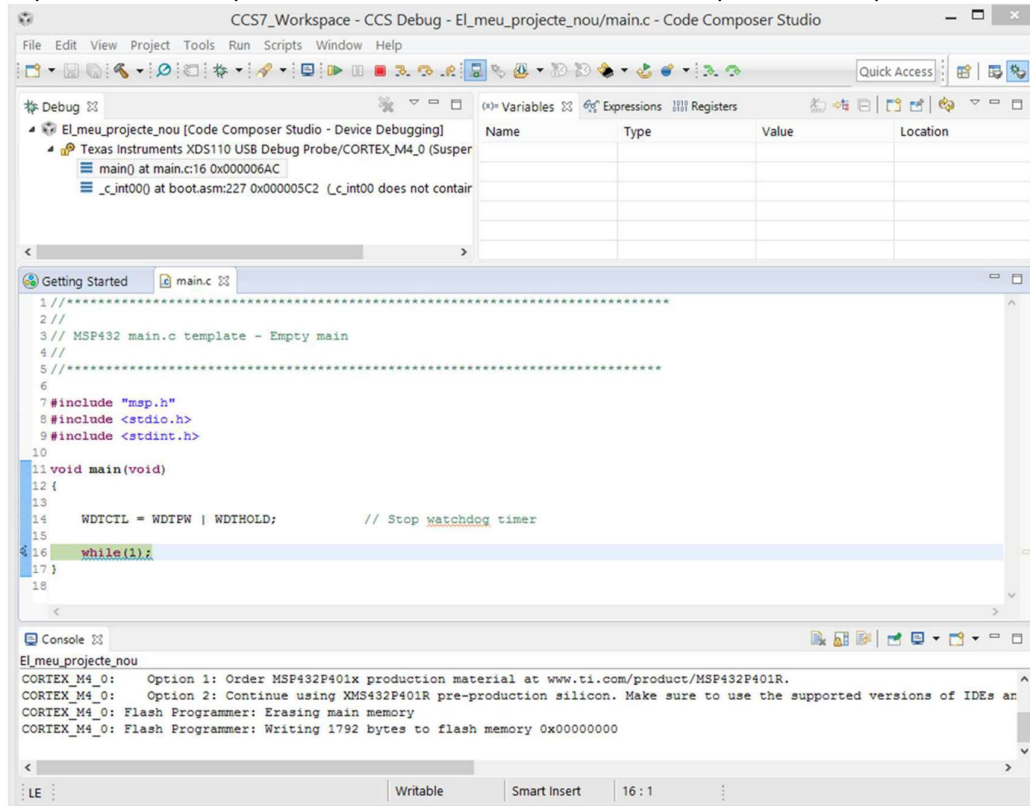


FIGURA 8 Vista de depuració del programa



En qualsevol moment podem tornar a la pantalla d'edició, i al contrari, mitjançant els botons que hi ha al cantó superior dret ("CCS Edit" i "CCS Debug").

A la pantalla de *debug* hi ha diferents finestres:

- Una primera que es diu **Debug** amb informació tècnica del projecte que s'està depurant.
- Una segona, que té varies pestanyes: "**Variables**" per visualitzar variables locals, "**Expressions**" per a qualsevol variable, tant local (si és accessible) com global, o qualsevol expressió combinant variables, i "**Registers**" per consultar i modificar registres de configuració del microcontrolador. Ens seran molt útils per veure els valors que van prenent les variables durant l'execució del programa. Compte que només es poden actualitzar quan el programa està aturat, per exemple l'hem posat **en pausa**, o durant **execució pas a pas** o quan arribem a un **breakpoint**.
- Una tercera finestra ens permet veure el codi dels arxius ".c" i ".h" que s'està executant. Aquí es podrà veure a quina instrucció ens trobem en cas d'execució pas a pas. També es pot modificar aquí el codi del programa, però si ho fem, l'haurem de tornar a carregar per a que els canvis es tinguin en compte.
- També ens apareixerà una finestra anomenada **Console** on s'informa de problemes durant l'execució.

INCLUSIÓ D'UN PROJECTE EXISTENT

Per tal de veure com funciona el depurador, farem servir un projecte ja existent.

En cas de voler importar un projecte el qual tenim en un carpeta del nostre ordinador, no s'ha de copiar directament en el workspace. El que hem de fer és seleccionar *Project -> Import CCS Project...*

A continuació, se'ns obre una nova finestra on pitjarem a **Browse** i triarem la carpeta on està el projecte, es a dir dintre del nostre *Workspace*. Si no hi és, primer ens el baixarem del campus virtual, el desarem i desempaquetarem a una carpeta temporal (D:\temp per exemple), i l'anirem a buscar en aquesta amb el **Browse**. En aquest cas però, haurem de marcar l'opció "*Copy projects into workspace*". Si ho hem fet bé, apareixerà en **Discovered Projects** el projecte que buscàvem.

El seleccionem i pitgem a **Finish**, veurem que a la finestra de projectes (*Project Explorer*) apareix el nou projecte. A més, es posarà com el projecte actiu (marcat amb negreta). Per definició, de projecte actiu només hi pot haver-ne un, i és sobre el que es fa la compilació i depuració. En qualsevol moment podem canviar el projecte actiu, només l'hem de seleccionar amb el ratolí a la mateixa finestra de projectes.

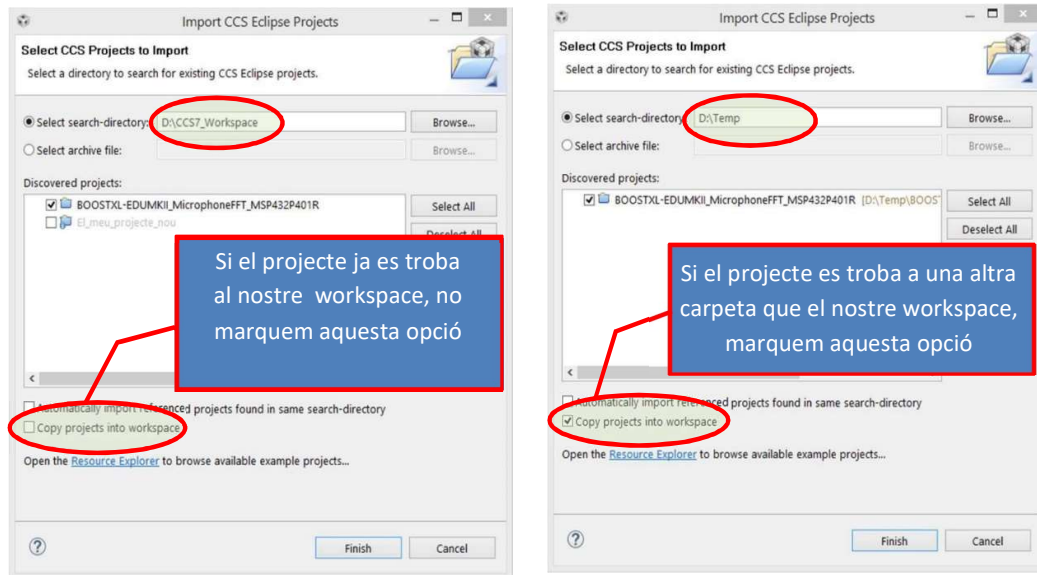


FIGURA 9 Importació d'un projecte de CCS extern


Selecció del projecte, podem veure els arxius del programa exemple de la placa d'experimentació. Veiem que al **main.c**, la primera instrucció de la funció **main()** és:

```
WDT_A->CTL = WDT_A_CTL_PW |          // Stop WDT
              WDT_A_CTL_HOLD;
```

que fa el mateix que la instrucció
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

que hem mencionat abans, però utilitzant estructures .

Es poden cercar elements del projecte, variables, llibreries, ports... a partir d'una paraula clau, i tenim dues formes:

- Mitjançant l'accés directe a la barra d'eines,  **Search**, i posant la paraula que cerquem al camp **Containing Text**, i fer **Search**.
- O bé, un cop seleccionada la paraula clau premem el botó dret del ratolí i al menú contextual també surt el **Search Text**. Podrem triar entre cercar a **Workspace**, al **Projecte** o només al **Fitxer** al que estem.

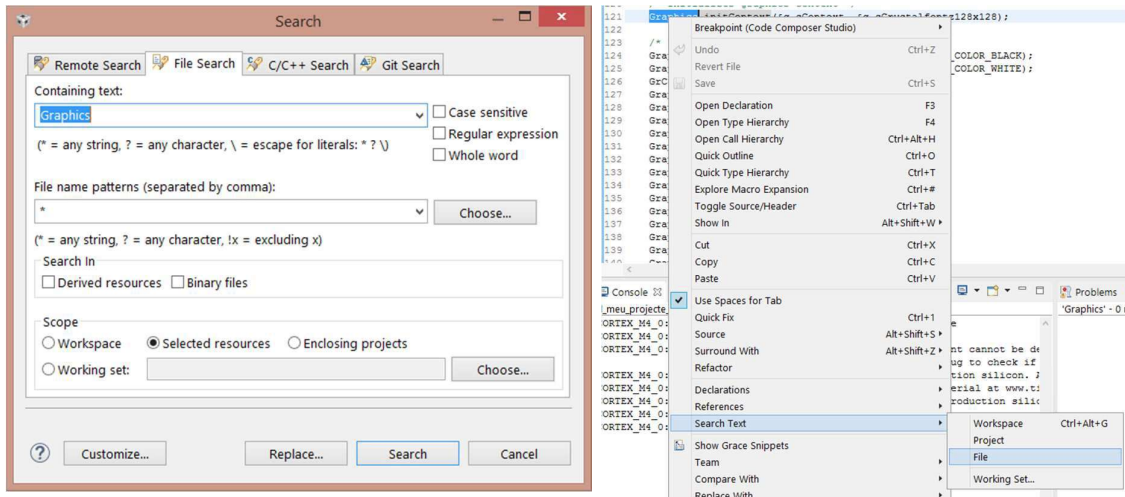


FIGURA 10 Formes de realitzar una cerca en el nostre projecte o workspace

Un cop feta la cerca, ens apareix una nova finestra, anomenada **Search**, a l'àrea inferior de la pantalla. En ella, apareixen tots els llocs on està el que nosaltres estem cercant.

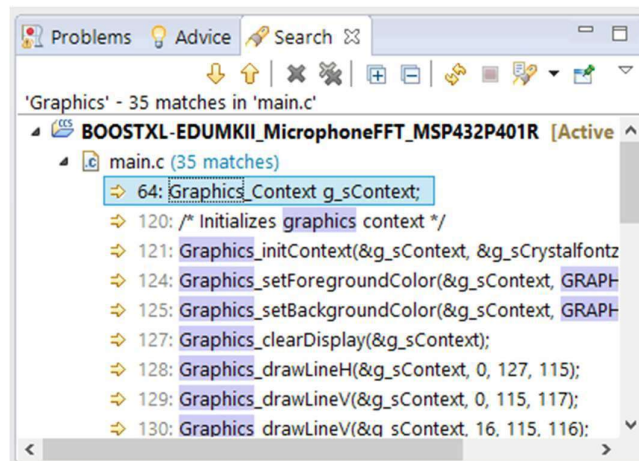



FIGURA 11 Resultat de la cerca

Fent doble clic en qualsevol dels resultats ens envia directament a la línia de codi corresponent. També disposem d'una sèrie de botons per gestionar els resultats, com moure's al resultat següent o anterior (fletxes), esborrar un resultat o tots (creus), etc...

DEPURACIO (DEBUG) PAS A PAS I BREAKPOINTS

Al depurar el projecte potser que ens doni errors. Els haurem de corregir tots abans de poder procedir a la fase de programació.

Un cop podem compilar el programa sense errors, el podem carregar al microcontrolador, per executar-lo i si és necessari depurar-lo (menú **Run -> Debug** o  de la Barra d'Eines)

La pantalla de depuració té dues barres d'eines, una semblant a la d'edició, que permet compilar (Build), carregar programes, cercar ... I una altra que és per a les tasques pròpies de depuració.

Ens dedicarem a aquesta:



Aquests accessos ràpids ens permeten controlar l'execució del programa.

Al primer bloc  els més interessants són:

- ***RUN*** que executa el programa a partir de la instrucció senyalada pel punter de la pantalla que mostra els fitxers.
- ***Suspend*** és una pausa que permet aturar l'execució, per continuar-la després, pas a pas si volem, des d'aquest punt.
- ***Terminate*** que para l'execució totalment i torna al mode d'edició.

El segon bloc permet l'execució pas a pas o de funció en funció:



Tant groc com verd, executen la següent instrucció, si és una funció entren en ella.



 Tant groc com verd, executa la següent instrucció, si és una funció la executa sencera (com si fos només una instrucció).



Per últim, si estem dintre d'una funció i volem executar seguit fins que s'acabi i sortim aquest és el nostre botó. Pot anar bé per acabar bucles o com hem dit funcions.

Amb les fletxes grogues, ens movem d'una instrucció de C a l'altra, i amb les verdes ens movem per micro-instruccions (assemblador).

BREAKPOINTS

L'execució pas a pas ens anirà molt bé perquè, com veurem després, ens permet controlar amb molta cura si el programa fa el que volem. Molts cops però volem executar en bloc un conjunt d'instruccions que ja sabem que funcionen bé fins a un punt que volem depurar pas a pas, o bé per conèixer l'estat del sistema a aquell punt. En aquestes situacions farem servir els punts de ruptura o **Breakpoints**.

Podem fer una prova al arxiu *main.c* del projecte que tenim obert. Suposem que volem introduir un Breakpoint just abans de que s'executi el toogle del LED. Ens posem sobre la línia just després del comentari (podeu fer una cerca per trobar-lo més ràpidament):

```
// Toggle P1.0 output
```

```
P1->OUT ^= BIT0;
```

```
// Blink P1.0 LED
```

i al menú contextual que surt amb el botó dret del ratolí en el marge de la línia on volem aturar el programa, seleccionem **Toggle Breakpoint**. Veurem que ens surt una mena de boleta a l'esquerra de la instrucció que indica que ara hi ha un breakpoint. També es pot afegir o suprimir un breakpoint fent doble clic directament en el marge a l'alcada de la línia que ens interessa.

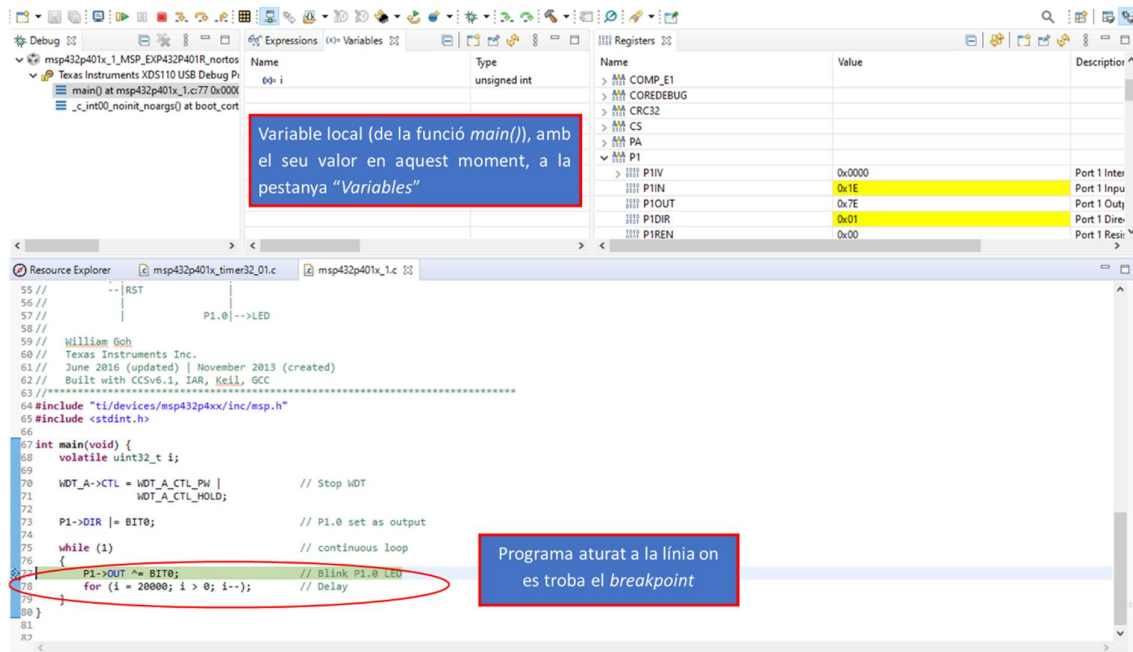


FIGURA 12 Programa aturat a un *breakpoint* i observació d'una variable a la finestra del *debugger*

Si ara executem el programa veurem que primer de tot parlarà en la primera línia dintre de la funció *main*. I si a continuació li donem a *Resume* (la fletxa verda) continuarà fins a parar-se en aquesta línia on hem afegit el breakpoint. A partir d'aquí podem continuar executant pas a pas, o simplement comprovar el valor d'alguna variable, port, registre, o paràmetre que ens interessi.

Si volem eliminar el *breakpoint*, hem de seguir el mateix procés, i tornar a seleccionar **Toggle Breakpoint** (o bé fent doble clic sobre la marca del mateix breakpoint).

OBSERVAR VARIABLES I ALTRES PARÀMETRES D'INTERÉS

Per ajudar-nos en la tasca de depuració, el programa ens proporciona una facilitat molt important que és el poder observar el valor que van prenent les variables, registres, ports, memòria...

Per exemple, a la finestra superior dreta veurem que hi ha una pestanya anomenada **Expressions**, ens fiquem a sobre de **Add New** i escrivim el nom de la variable de la qual volem saber el valor, o alguna expressió que volem que s'avalui. Un altre opció és seleccionar la variable al programa i al menú contextual del botó dret del ratolí seleccionar **Add Watch Expression**.

S'ha de tenir en compte que només podrem veure els valors quan el programa estigui aturat, bé perquè estem a un *breakpoint*, bé perquè estem fent una execució pas a pas.

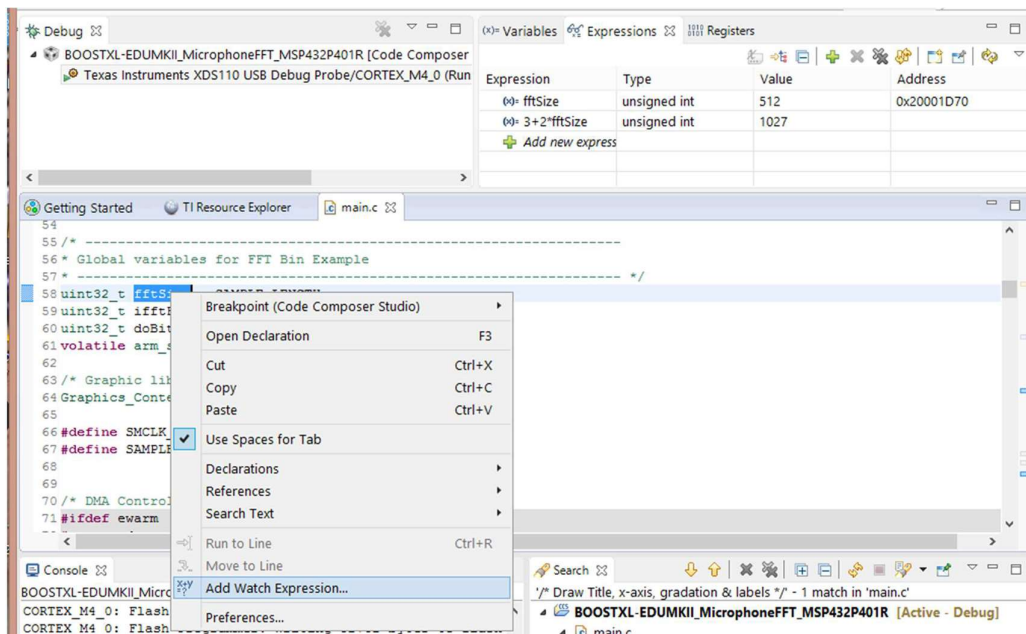


FIGURA 13 Exemple d'afegir una variable en la llista d'expressions a avaluar

Podem canviar el format amb el que veiem el valor entre diferents opcions: número natural, binari, decimal, o hexadecimal.

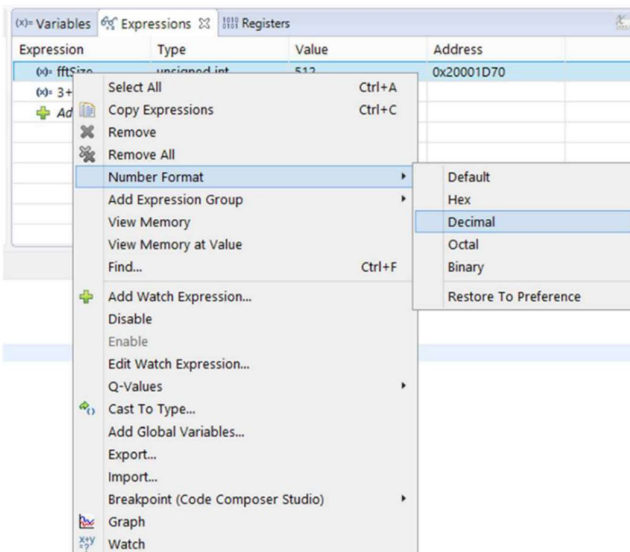


FIGURA 14 Exemple de canviar el format en el qual visualitzar el resultat d'una expressió

Podem veure altres elements com Registres i Ports, Memòria... Per fer això hem d'anar a *Window -> Show View -> Other*. Se'n obrirà una finestra i hem d'entrar a **Debug**. Aquí triarem què és el que volem veure.

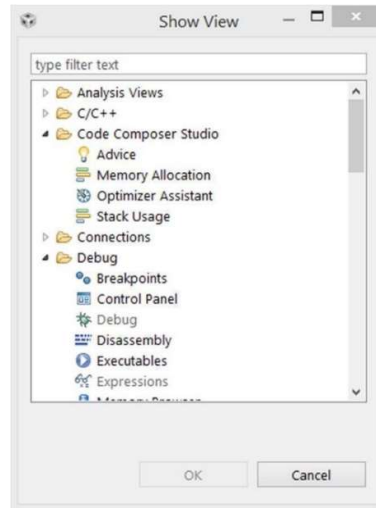


FIGURA 15 Altres possibles finestres a visualitzar

COMBINACIONS DE TECLES ÚTILS:

- **CTRL + SHIFT + F** : Permet formatar (i.e. tabular i organitzar el codi) correctament seguint la guia d'estils seleccionada.
- **CTRL + I** : per formatar amb indentació un bloc de codi seleccionat -> molt útil per veure millor la jerarquia d'un programa (i detectar errors quan falta alguna "}", per exemple!).
- **CTRL + Espai** : Auto-completar. Comenceu a escriure el nom d'una variable, funció, estructura en C (for, if, etc) , etc i ús permetrà saber les opcions disponibles amb els caràcters escrits.
- **CTRL + SHIFT + /** : per comentar / descomentar ràpidament un bloc de codi seleccionat
- **CTRL + Clic Esquerra** sobre una Funció, variable, define, ... : permet saltar a la declaració d'aquest (encara que es trobi dintre d'un altre arxiu del projecte). Després, amb Alt + fletxa esquerra (o la fletxa enrere groga de la barra d'eines) permet tornar al punt inicial.

EXERCICI PROPOSAT

En els microcontroladors, depenent de l'arquitectura d'aquests i el compilador utilitzat, ens podem trobar que una variable de tipus *int* tingui mides màximes diferents. Per evitar problemes de portabilitat del codi i tenir un millor control de la memòria utilitzada en les aplicacions encastades que solen tenir unes limitacions més significatives, és una bona pràctica emprar sempre la llibreria *stdint.h*. Aquesta llibreria ens permet fer servir enters amb o sense signe d'una mida, en bits, fixa per qualsevol entorn. Per exemple, si necessitem una variable comptador que sabem mai podrà ser major de 200, podem fer servir una variable del tipus `uint8_t` ($2^8 - 1 = 255$ de valor màxim). En canvi, si sabem que hi ha un valor que té signe i pot arribar a ser molt gran, podem utilitzar una variable `int64_t` ($2^{63} - 1 \approx 9e-18$, en aquest cas és 63 donat que perdem un bit de signe). Un altre exemple és el cas del conegut tipus "*char*": sempre serà de 8 bits, però segons el compilador, pot ser amb signe o sense signe!

Creeu un projecte nou, i afegiu el següent codi en el vostre projecte. Executeu-lo pas a pas per veure com canvien els valors:

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

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

    volatile uint64_t foo;
    uint8_t a, c = 0, i, d;
    int8_t b;
    char string[16];

    d = 5;

    a = 255;
    b = a;
    a = 0x11FF;

    a = 0x01;
    a <<= 1;
    a = (0x10 >> 1);

    a = 0x80;
    a <<= 1;
    a = 0xFF;
    a >>= 1;
    b = 0xFF;
    b >>= 1;

    a = 0xAA;
    a ^= 0xFF;
    a ^= 0xFF;
```



```
a &= 0x0F;
a &= ~(0x01);

a |= 0x0F;
a |= ~(0xFE);

string[0] = 32;
for (i = 1; i < sizeof(string); ++i) {
    string[i] = ' ';
}

sprintf(string, "Hello world!");
sprintf(string, "Hello!");
a = 101;
sprintf(string, "Hello %d", a);
a = 11;
sprintf(string, "Hello %d", a);
foo = sizeof(string);
sprintf(string, "Hello %3d", a);
a = 101;
sprintf(string, "Hello world %03d!", a);
foo = sizeof(string);

while(1) {
    __no_operation();
}
}
```

Analitzeu el codi i fixeu-vos bé en els warnings del compilador i enteneu-los. En cas que no tingueu clar l'afecte, analitzeu que passa amb les variables aquestes al executar-se pas a pas.

Per aquesta pràctica no cal entregar cap informe. Igualment, reviseu en l'apartat documentació del campus com crear correctament els informes.

En les entregues, haureu de pujar sempre tant l'informe com el projecte del code composar. Per tal d'estalviar espai i evitar problemes quan compartiu amb els professors el vostre projecte, abans de copiar el vostre projecte, feu clic dret en aquest dintre de la finestra del *Project Explorer* i seleccioneu l'opció *Clean Project*.