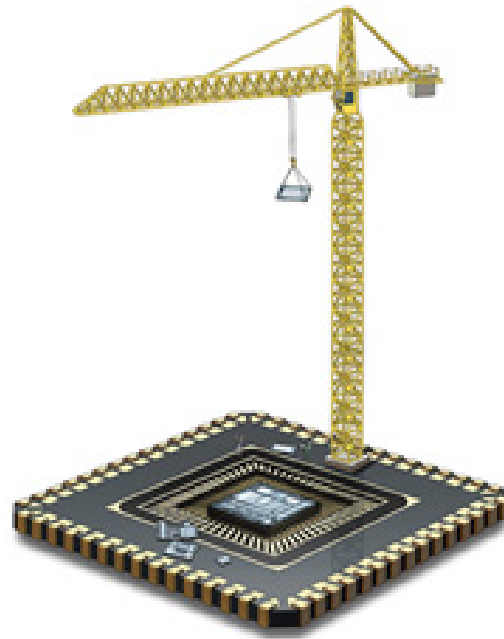


Outil de développement *IAR Embedded Workbench IDE*



Introduction

L'outil de développement intégrés « **IAR Embedded Workbench™** » est un environnement de développement intégrés très puissant (IDE : Integrated Development Environment), permettant de développer et gérer applications embarquées sous forme de projets.

C'est une plate-forme de développement, avec toutes les configurations nécessaires pour des ingénieurs de développement.

Tous les outils nécessaires aux développements ont été intégrés dans l'environnement de travail, soit:

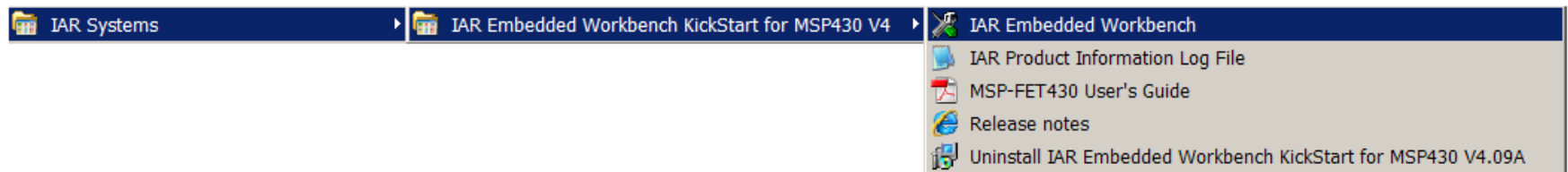
- ⇒ un compilateur C/C++ optimisé pour les applications utilisant un MSP430,
- ⇒ un assembleur pour le MSP430,
- ⇒ un éditeur de lien XLINK Linker™,
- ⇒ une bibliothèque Builder™ d'IAR XAR,
- ⇒ un éditeur convivial,
- ⇒ un système de management de projet,
- ⇒ un programme de mise au point (debugger) IAR C-SPY™.

Création d'un projet

Ouvrir l'outil de développement intégrés « **IAR Embedded Workbench™** » en sélectionnant l'icône placée sur le Bureau

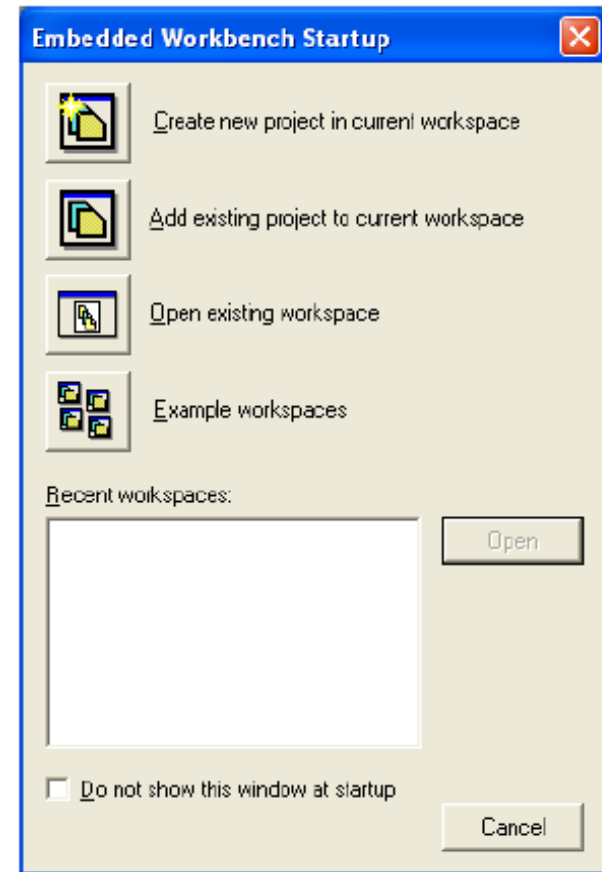


Ou en passant le menu déroulant des programmes



Création d'un projet

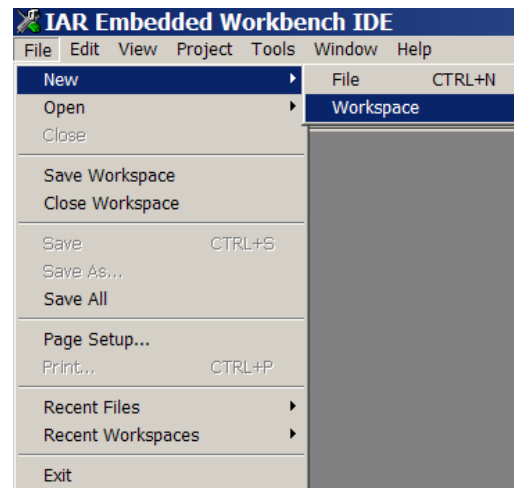
Au démarrage la fenêtre suivante apparaît



Cliquer sur « Cancel » pour fermer cette fenêtre

Création d'un projet

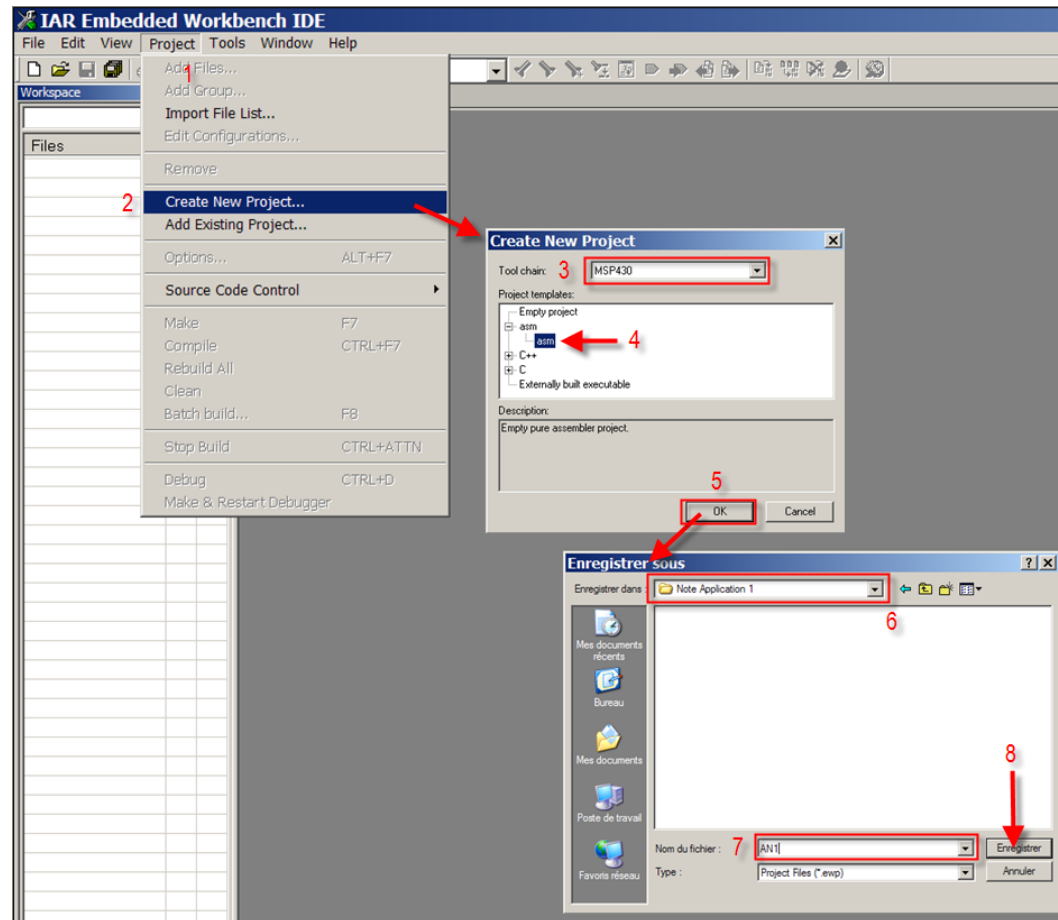
Sélectionner File → New → Workspace



Un espace de travail vierge apparaît

Création d'un projet en assembleur

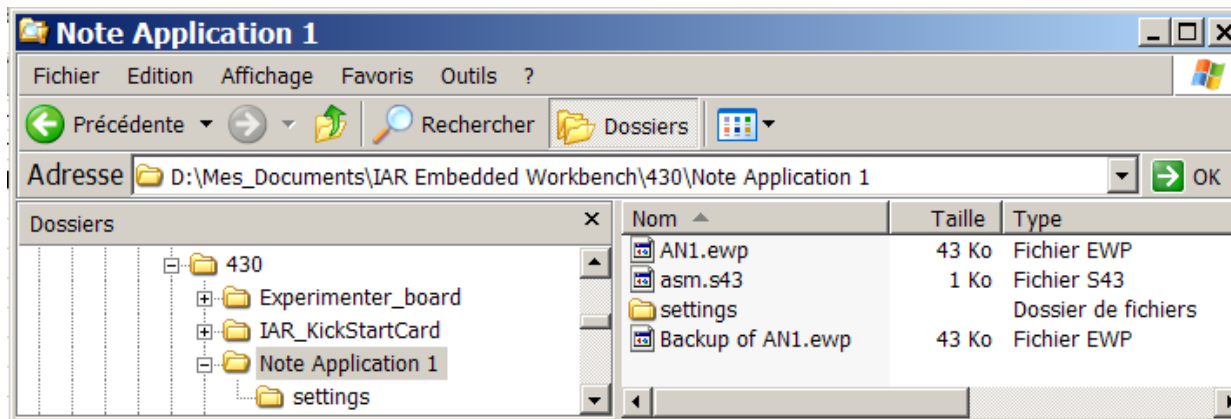
1. Pour créer un nouveau projet, sélectionner **Project**
2. Dans le menu déroulant, sélectionner **Create New Project**
3. S'assurer que la boîte de dialogue « Tool chain » indique « MSP430 ».
4. Choisir un projet assembleur asm
5. Cliquer sur ok.
6. Dans la nouvelle fenêtre choisir le dossier crée pour ce projet ou créer un nouveau dossier dans l'emplacement prévu.
7. Donner un nom au projet.
8. Enregistrer le projet.



Création d'un projet en assembleur

Un sous-dossier setting a été créé dans le dossier racine Note Application 1 du projet.
Ce sous-dossier contient les fichiers suivants :

- AN1.dep : contient les informations sur la dépendance entre les divers fichiers sources, objets, exécutable, ...
- AN1.ewp : contient les indications propres à un projet compatible IAR EW
- asm.s43 : contient le programme source
- Backup of AN1.ewp : même contenu que le fichier AN1.ewp



Création d'un projet en assembleur

Le projet apparaît dans l'espace de travail.

Par défaut deux configurations de projet sont créées : « Debug » et « Release ».

Pour ce tutorial, seul la configuration « Debug » est utilisée.

Le choix entre ces deux configurations se fait à partir du menu déroulant dans la partie supérieure de la fenêtre.

Le vu (✓ : 1) situé en face du nom du projet indique que ce projet est actif.

L'astérisque rouge (* : 2) à la suite du nom du fichier source asm.s43 indique que le fichier objet correspondant n'existe pas ou n'est pas à jour.

Les fichiers visibles dans l'espace de travail sont :

- ⇒ asm.s43 : fichiers contenant le programme assembleur source
- ⇒ asm.r43 : fichier objet
- ⇒ asm.d43 : fichier exécutable

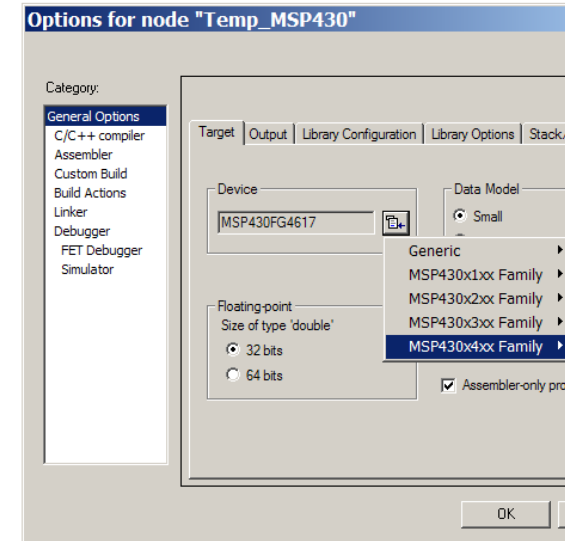
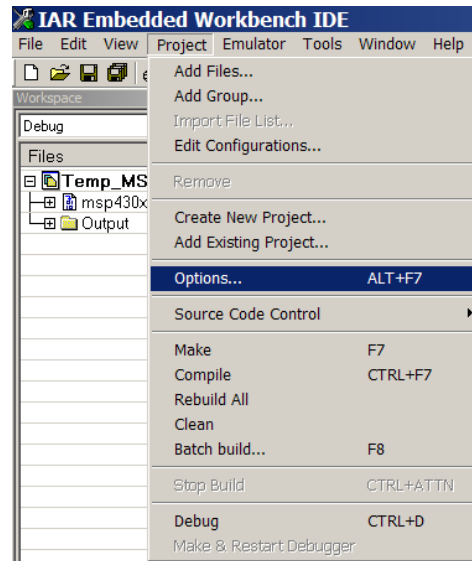
Le contenu du fichier source asm.s43, généré automatiquement lors de la création du projet contient du code assembleur générique pour une application de base.

Création d'un projet en assembleur

Options du projet

Les options appliqués au projet en cours permettent de fixer des paramètres comme :

Le type du microcontrôleur (dans notre cas MSP430F2012, MSP430F2013, MSP430FG4617, selon la carte utilisée)



- MSP430C412
- MSP430C413
- MSP430F412
- MSP430F413
- MSP430F415
- MSP430F417
- MSP430F423
- MSP430F423A
- MSP430F425
- MSP430F425A
- MSP430F426
- MSP430F427
- MSP430F427A
- MSP430F435
- MSP430F4351
- MSP430F436
- MSP430F4361
- MSP430F437
- MSP430F4371
- MSP430F447
- MSP430F448
- MSP430F449
- MSP430F4783
- MSP430F4784
- MSP430F4793
- MSP430F4794
- MSP430FE423
- MSP430FE423A
- MSP430FE425
- MSP430FE425A
- MSP430FE427
- MSP430FE427A
- MSP430FG4250
- MSP430FG4260
- MSP430FG4270
- MSP430FG437
- MSP430FG438
- MSP430FG439
- MSP430FG4616
- MSP430FG4617**
- MSP430FG4618
- MSP430FG4619
- MSP430FW423
- MSP430FW425
- MSP430FW427

Création d'un projet en assembleur

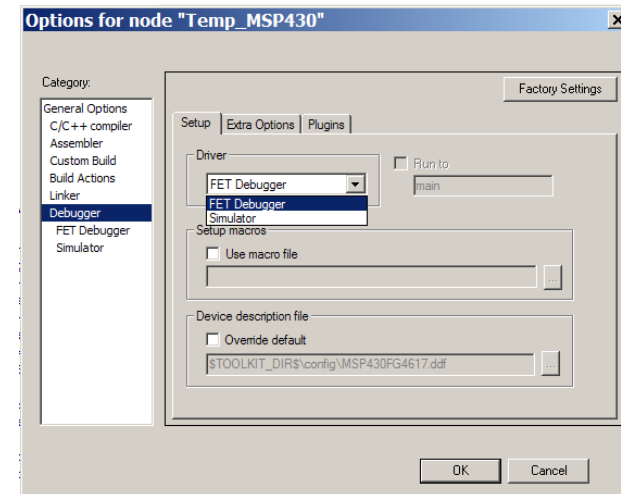
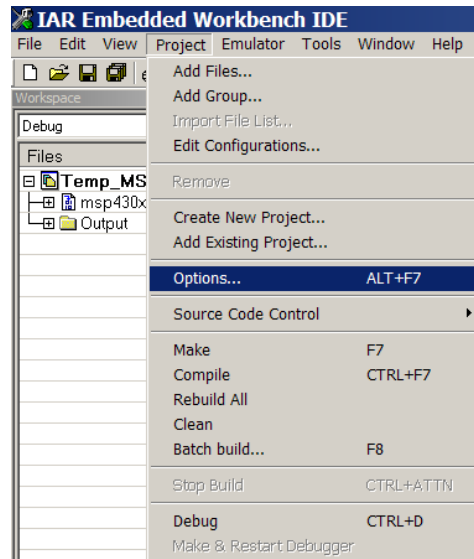
Options du projet

Les options appliqués au projet en cours permettent de fixer des paramètres comme :

Le type du debugger, c'est-à-dire :

le simulateur intégré

l'interface de debug externe



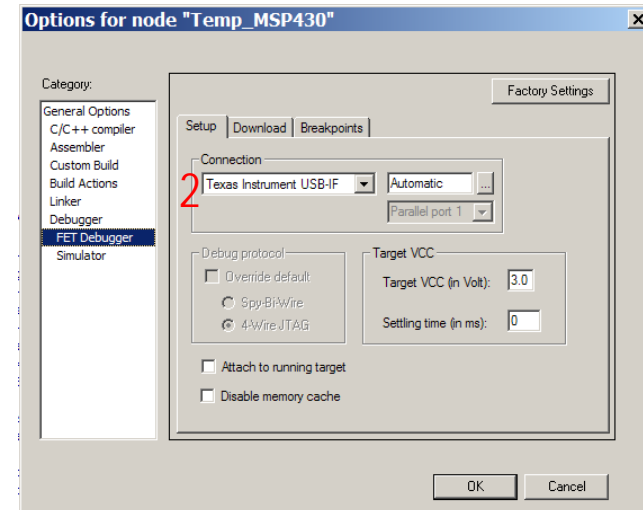
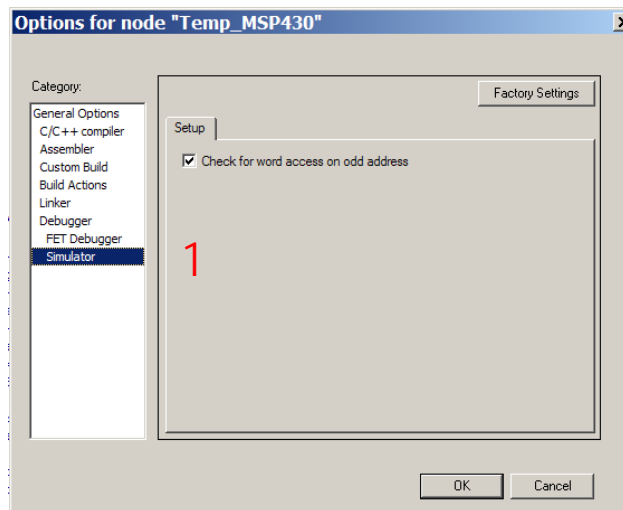
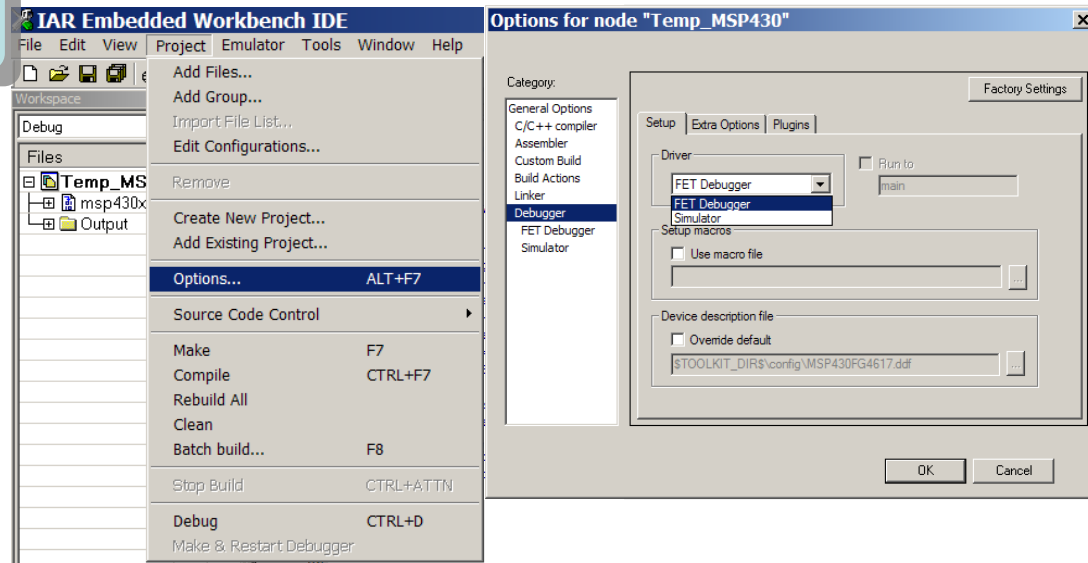
Création d'un projet en assembleur

Options du projet

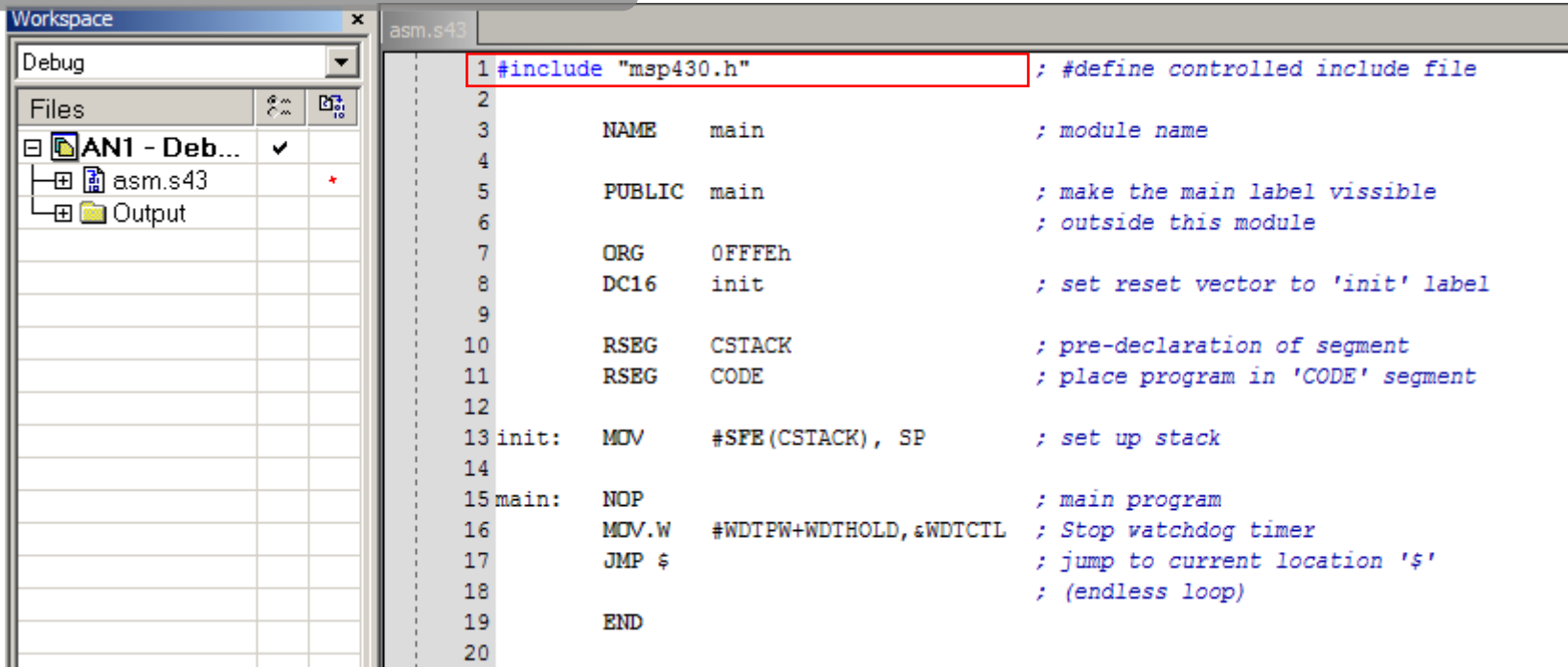
Les options appliqués au projet en cours permettent de fixer des paramètres comme :

Le type du debugger, c'est-à-dire :

1. le simulateur intégré
2. l'interface de debug externe



Création d'un projet en assembleur



```
Workspace x
asm.s43
1 #include "msp430.h" ; #define controlled include file
2
3     NAME    main      ; module name
4
5     PUBLIC  main      ; make the main label visible
6                   ; outside this module
7
8     ORG    0FFFEh     ;
9
10    RSEG   CSTACK     ; pre-declaration of segment
11    RSEG   CODE       ; place program in 'CODE' segment
12
13init:  MOV    #SFE(CSTACK), SP ; set up stack
14
15main:  NOP           ; main program
16      MOV.W  #WDTPW+WDTHOLD, &WDTCTL ; Stop watchdog timer
17      JMP   $         ; jump to current location '$'
18                   ; (endless loop)
19
20    END
```

#include "msp430.h"

Directive permettant d'inclure le fichier d'entête msp430.h.

if ...

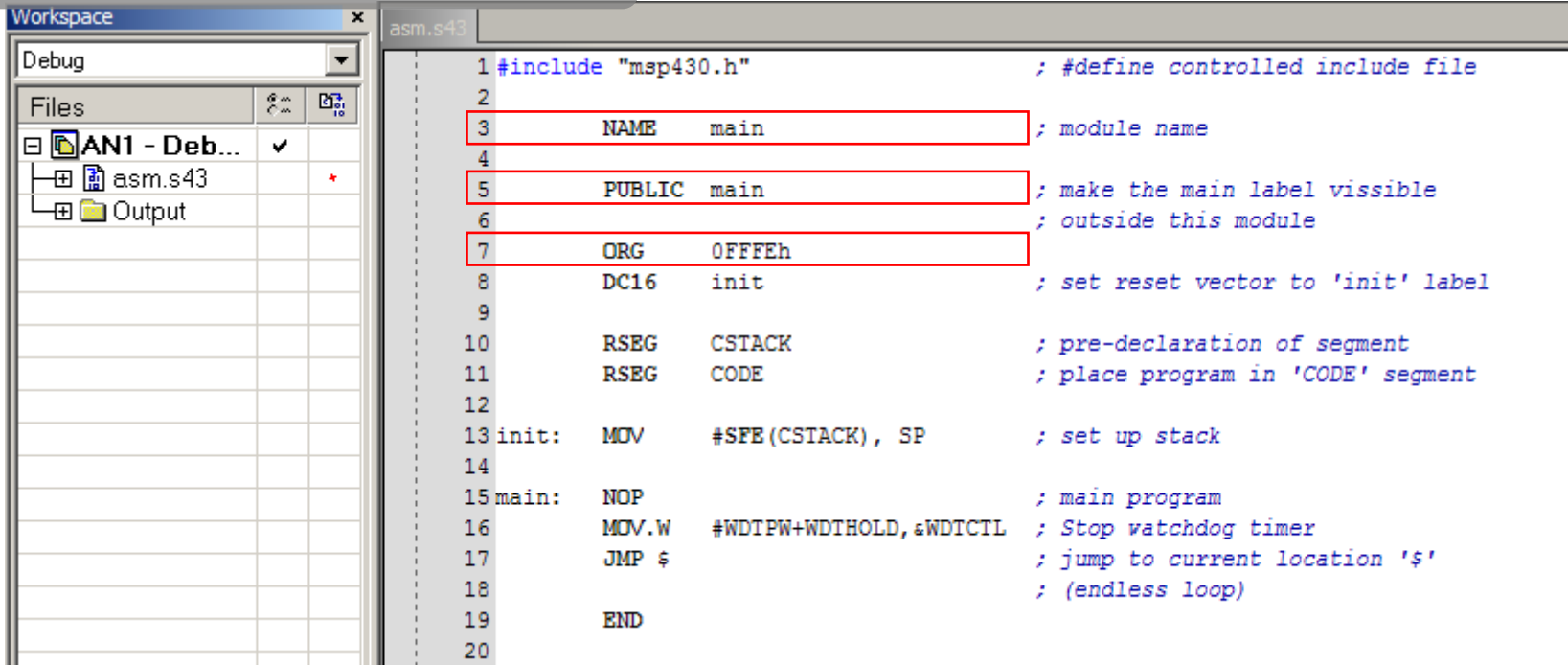
```
#elif defined (__MPS430FG4616__) || defined (__MPS430FG4617__) || defined (__MSP430FG4618__)
```

```
#include "msp430xG46x.h"
```

Ce fichier contient une suite de tests permettant d'inclure le fichier d'entête du MSP430 défini dans :

Projet → Option → General Options → Target → Device, soit le MSP430FG4617

Création d'un projet en assembleur



```
1 #include "msp430.h"           ; #define controlled include file
2
3  NAME    main                 ; module name
4
5  PUBLIC  main                 ; make the main label visible
6                               ; outside this module
7  ORG    0FFFEh                ;
8  DC16   init                  ; set reset vector to 'init' label
9
10 RSEG   CSTACK                 ; pre-declaration of segment
11 RSEG   CODE                   ; place program in 'CODE' segment
12
13 init:  MOV    #SFE(CSTACK), SP ; set up stack
14
15 main:  NOP                    ; main program
16        MOV.W #WDTPW+WDTHOLD, &WDTCTL ; Stop watchdog timer
17        JMP  $                  ; jump to current location '$'
18                               ; (endless loop)
19
20  END
```

NAME main

Définit le début d'un module (programme) dont le nom est **main**

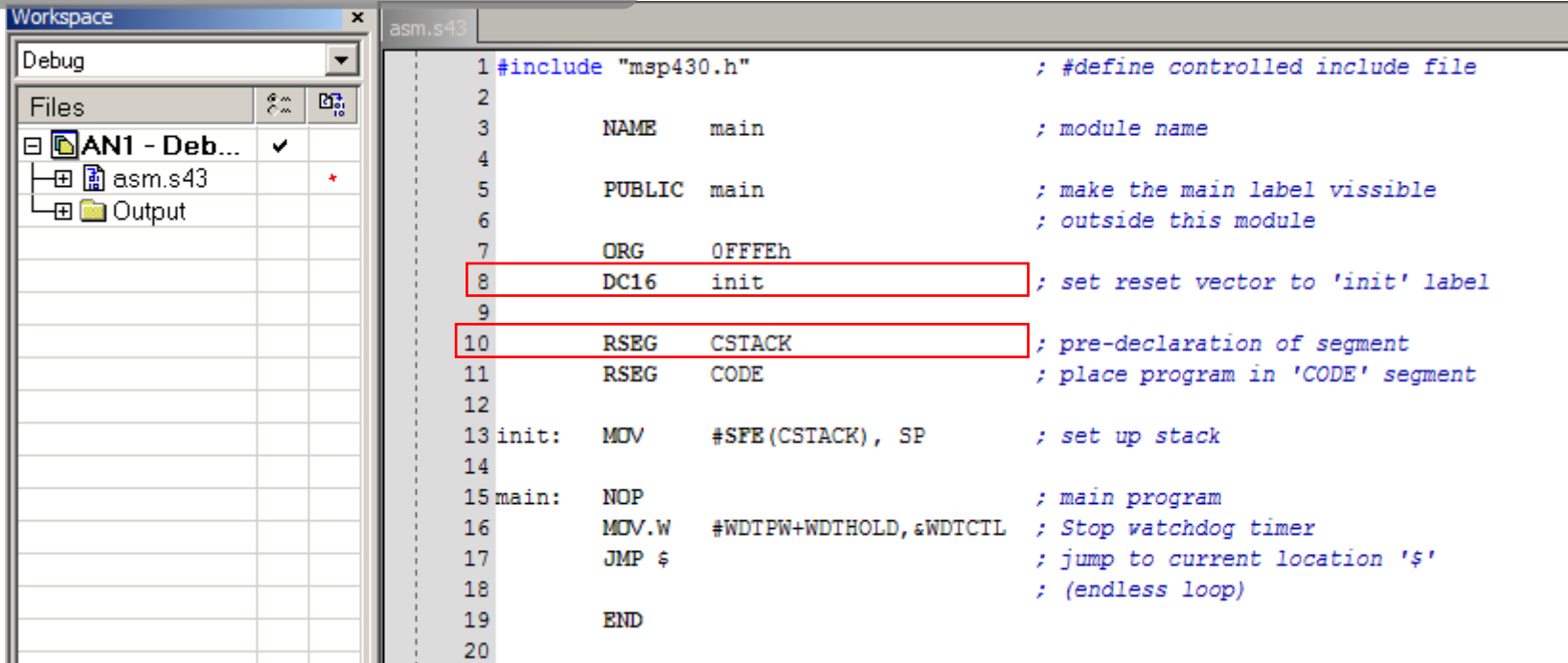
PUBLIC main

Rend l'étiquette **main** disponible pour d'autres modules (global)

ORG 0FFFEH

Force le compteur de programme de section à la valeur 0FFFEH, soit l'adresse du vecteur d'interruption RESET

Création d'un projet en assembleur



```
Workspace
Debug
Files
AN1 - Deb...
  asm.s43
  Output

asm.s43
1 #include "msp430.h"           ; #define controlled include file
2
3     NAME    main              ; module name
4
5     PUBLIC  main              ; make the main label visible
6                               ; outside this module
7
8     ORG     0FFFFh            ; set reset vector to 'init' label
9
10    RSEG    CSTACK             ; pre-declaration of segment
11    RSEG    CODE               ; place program in 'CODE' segment
12
13 init:    MOV     #SFE(CSTACK), SP    ; set up stack
14
15 main:    NOP                   ; main program
16          MOV.W  #WDTPW+WDTHOLD, &WDTCTL ; Stop watchdog timer
17          JMP    $               ; jump to current location '$'
18                               ; (endless loop)
19
20    END
```

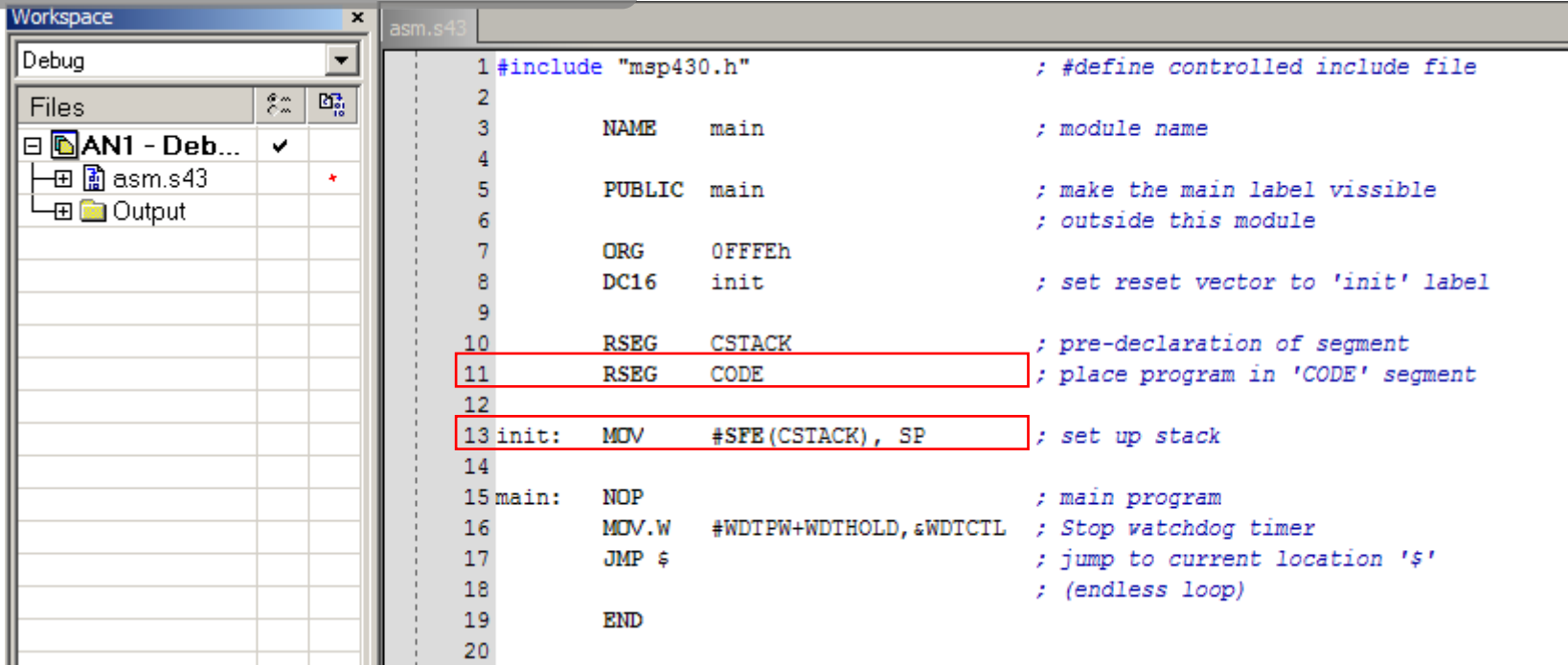
DC16 init

Place à l'adresse **init** (adresse de début du programme) dans zone de mémoire réservée aux vecteurs d'interruption, plus précisément à l'adresse correspondant à l'interruption non masquable (NMI) nommée RESET

RSEG CSTACK

Défini une nouvelle section nommée **CSTACK** qui permet de réserver une zone mémoire à la pile. Cette section est réadressable (relocatable) lors du lien entre tous les fichiers objets (éditeur de lien).

Création d'un projet en assembleur



```
1 #include "msp430.h"           ; #define controlled include file
2
3     NAME    main              ; module name
4
5     PUBLIC  main              ; make the main label visible
6                               ; outside this module
7
8     ORG     0FFFEh            ; set reset vector to 'init' label
9
10    RSEG    CSTACK             ; pre-declaration of segment
11    RSEG    CODE               ; place program in 'CODE' segment
12
13 init:    MOV    #SFE(CSTACK), SP ; set up stack
14
15 main:    NOP                  ; main program
16          MOV.W  #WDTPW+WDTHOLD, &WDTCTL ; Stop watchdog timer
17          JMP    $              ; jump to current location '$'
18                               ; (endless loop)
19
20    END
```

REG CODE

Défini une nouvelle section nommée **CODE** qui permet de réserver une zone mémoire pour la partie programme (code). Cette section est réadressable (relocatable) lors du lien entre tous les fichiers objets (éditeur de lien)

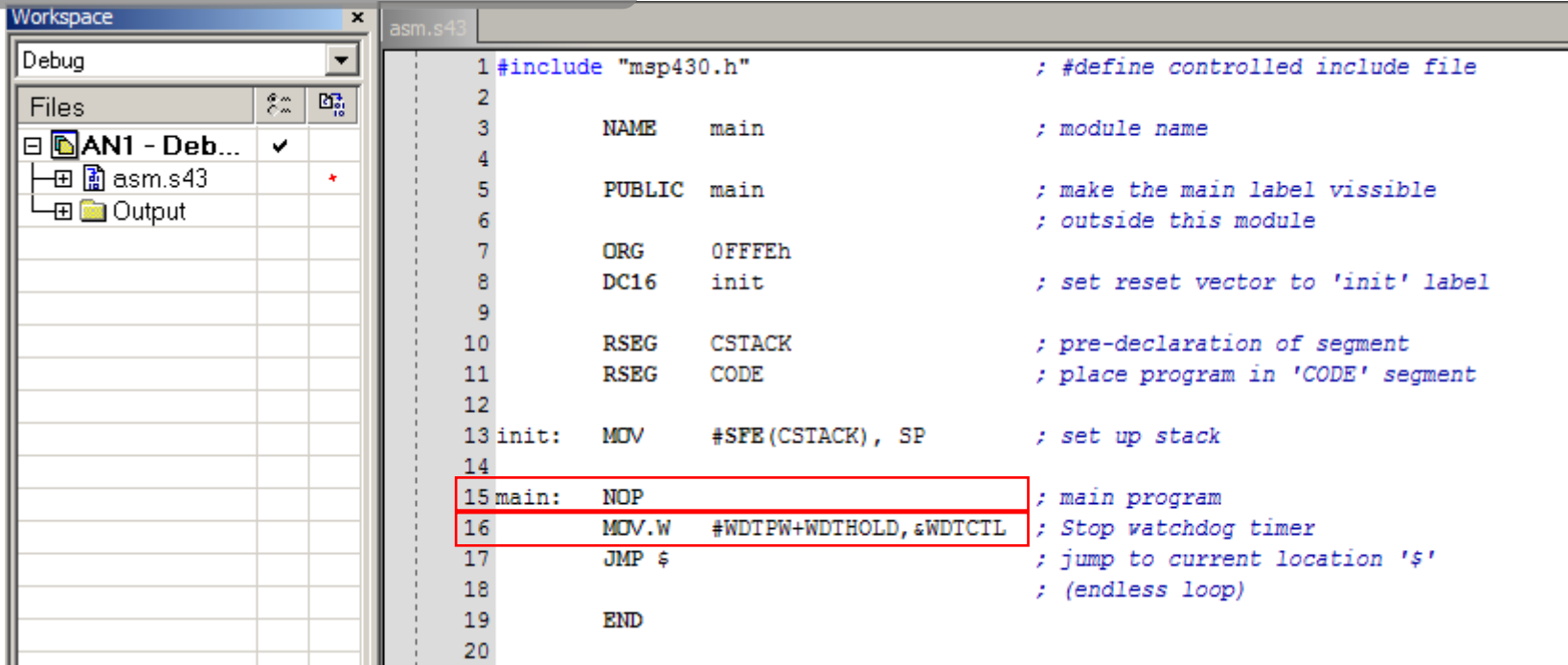
init: MOV #SFE(CSTACK), SP

#SFE(CSTACK) donne l'adresse de fin du segment dans le nom se trouve dans la parenthèse (CSTACK). Cette instruction permet d'initialiser la pile (stack)

main: NOP

Cette première instruction effectue aucune opération

Création d'un projet en assembleur



```
1 #include "msp430.h"           ; #define controlled include file
2
3     NAME    main              ; module name
4
5     PUBLIC main              ; make the main label visible
6                               ; outside this module
7
8     ORG    0xFFFEh           ; set reset vector to 'init' label
9
10    RSEG   CSTACK             ; pre-declaration of segment
11    RSEG   CODE               ; place program in 'CODE' segment
12
13 init:  MOV    #SFE(CSTACK), SP ; set up stack
14
15 main:  NOP                   ; main program
16        MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer
17        JMP  $                ; jump to current location '$'
18                               ; (endless loop)
19
20    END
```

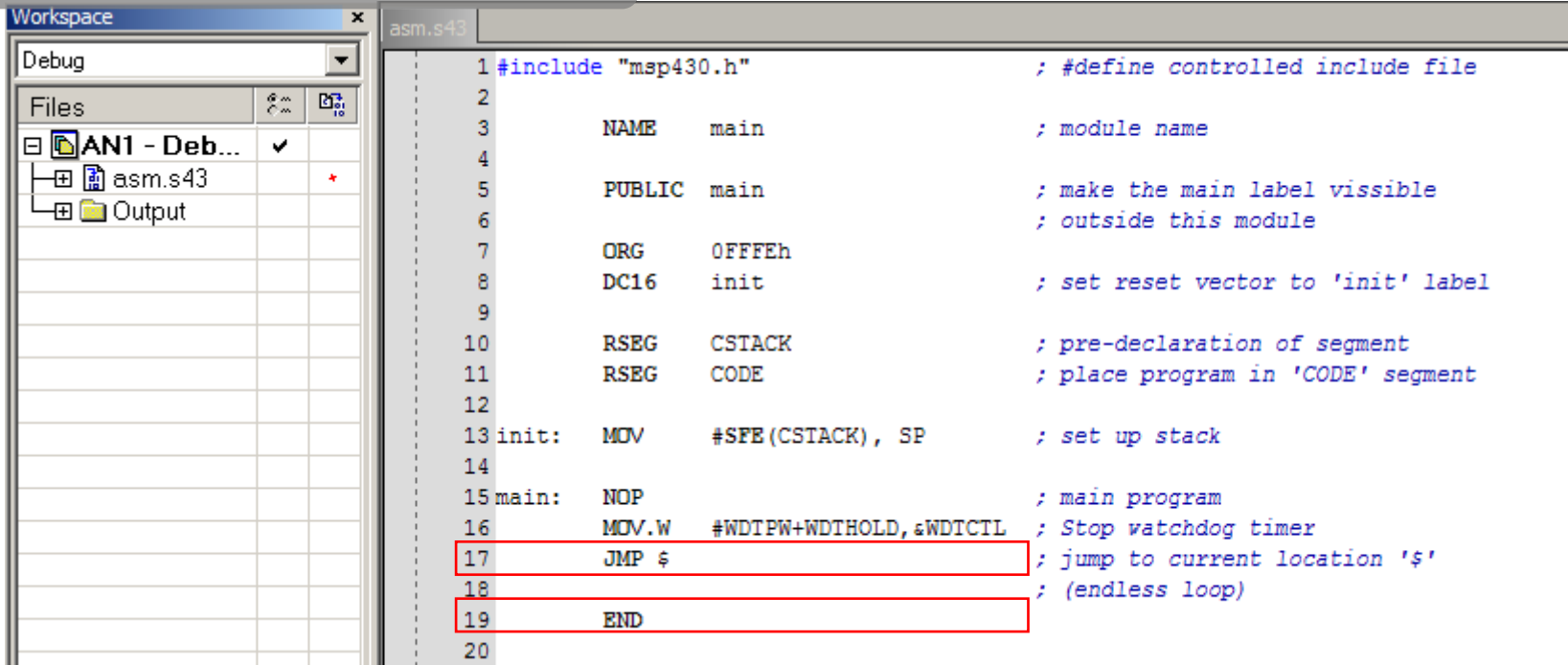
main: NOP

Cette première instruction effectue aucune opération

MOV.W #WDTPW+WDTHOLD,&WDTCTL

Cette instruction permet de désactiver le watchdog. Pour plus de détails se référer à la périphérie en question

Création d'un projet en assembleur



```
1 #include "msp430.h"           ; #define controlled include file
2
3     NAME    main              ; module name
4
5     PUBLIC  main              ; make the main label visible
6                               ; outside this module
7
8     ORG    0FFFEh             ; set reset vector to 'init' label
9
10    RSEG   CSTACK              ; pre-declaration of segment
11    RSEG   CODE                ; place program in 'CODE' segment
12
13 init:  MOV    #SFE(CSTACK), SP ; set up stack
14
15 main:  NOP                    ; main program
16        MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer
17        JMP  $                  ; jump to current location '$'
18
19        END
20
```

JMP \$

Instruction de saut à l'adresse courant. Cette instruction est nécessaire puisque elle évite que la suite aléatoire des instructions de la zone mémoire non attribuée soient exécutées.

END

Indique la fin du module (programme) MAIN

Espace de travail

Programme source

Contenu de la mémoire de programme

Visualisation des registres

The screenshot displays the IAR Embedded Workbench IDE interface. The main window is divided into three panes:

- Source Code Pane:** Shows assembly code for a program named 'main'. Key lines include:

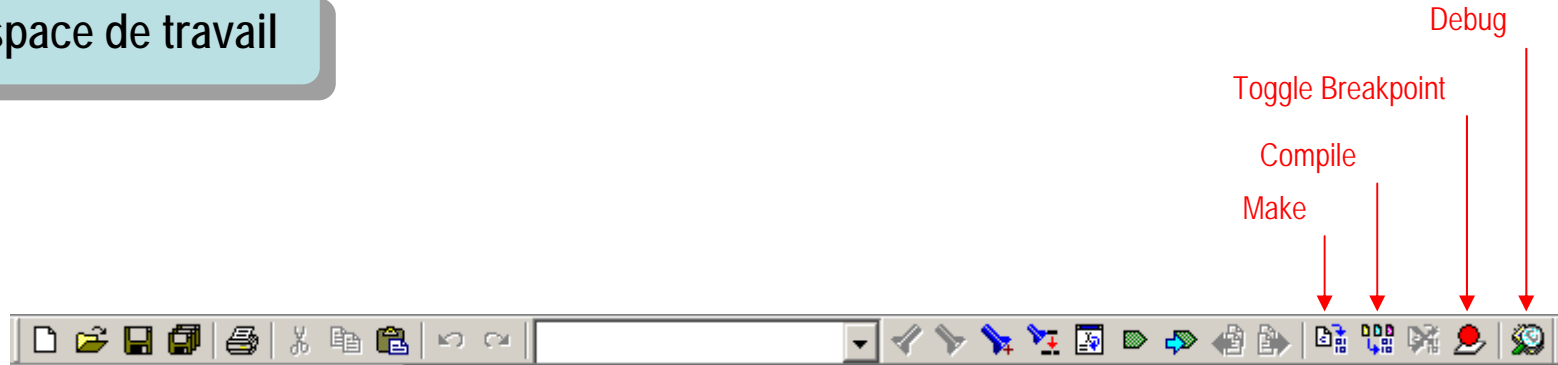
```
1 #include "map430.h" ; #define controlled include file
2
3 NAME main ; module name
4
5 PUBLIC main ; make the main label visible
6 ; outside this module
7 ORG 0FFFEH init ; set reset vector to 'init' label
8 DC16 init
9
10 RSBG CSTACK ; pre-declaration of segment
11 RSBG CODE ; place program in 'CODE' segment
12
13 init: MOV #SP(CSTACK), SP ; set up stack
14
15 main: NOP ; main program
16 MOV.W #WDTPW+WDTHOLD,&WDTC1L ; Stop watchdog timer
17 JMP $ ; jump to current location '$'
18 ; (endless loop)
19
20 END
```
- Memory Content Pane:** Shows the disassembled memory content. The first few lines are:

```
init: 001100 4031 0A00 mov.w #0xA00,SP
main: NOP ; main program
001104 4303 nop
MOV.W #WDTPW+WDTHOLD,&WDTC1L ; Stop watchdog timer
001106 40B2 5A80 0120 mov.w #0x5A80,&WDTC1L ; jump to current location '$'
```
- Register Pane:** Shows the status of CPU registers. The registers are listed with their current values:

Register	Value
PC	= 0x1100
SP	= 0x0000
SR	= 0x0000
R4	= 0x6296
R5	= 0x3260
R6	= 0x4262
R7	= 0x5AE7
R8	= 0x23AA
R9	= 0x0E9E
R10	= 0x497C
R11	= 0x035F
R12	= 0x3D30
R13	= 0x2481
R14	= 0x0EB0
R15	= 0x146F
CYCLECOUNTER	= 0
CCTIMER1	= 0
CCTIMER2	= 0
CCSTEP	= 0

The bottom status bar shows the current project configuration: AN1 - Debug. The Messages pane at the bottom left displays the build process output, including the command used to build the program.

Espace de travail



Make : mise à jour du fichier exécutable en compilant, assemblant et liant seulement les fichiers qui ont changé depuis la dernière compilation

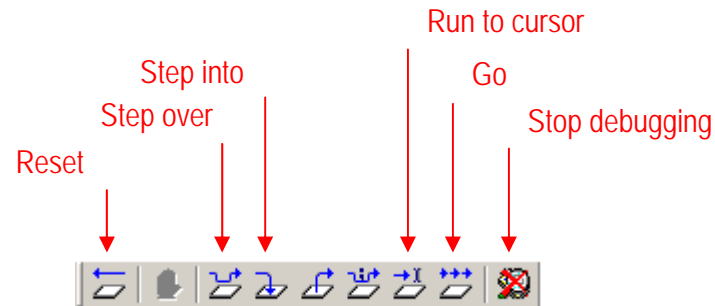
Compile : compile ou assemble un fichier source particulier. Le fichier est sélectionné soit dans la zone de travail (workspace), soit dans la zone d'édition

Toogle Breakpoint : Change l'état du point d'arrêt désigné par le curseur dans la fenêtre d'édition du programme

Debug : même action que Make mais chargement du programme sur la cible et activation des fonctions de debuggage

Espace de travail

Les icones suivantes permettent de contrôler le debuggage du programme



Reset : retour au début du programme

Step over : pas à pas sans introduction dans les sous routines

Step into : pas à pas avec introduction dans les sous routines

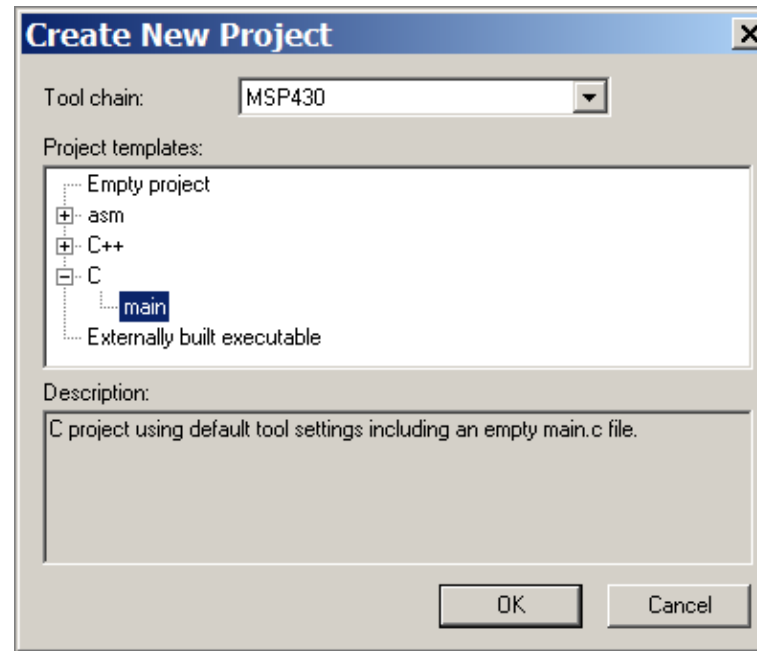
Run to cursor : exécution jusqu'à la position du curseur

Go : exécution jusqu'au prochain « breakpoint »

Stop debugging : arrêt du mode debug

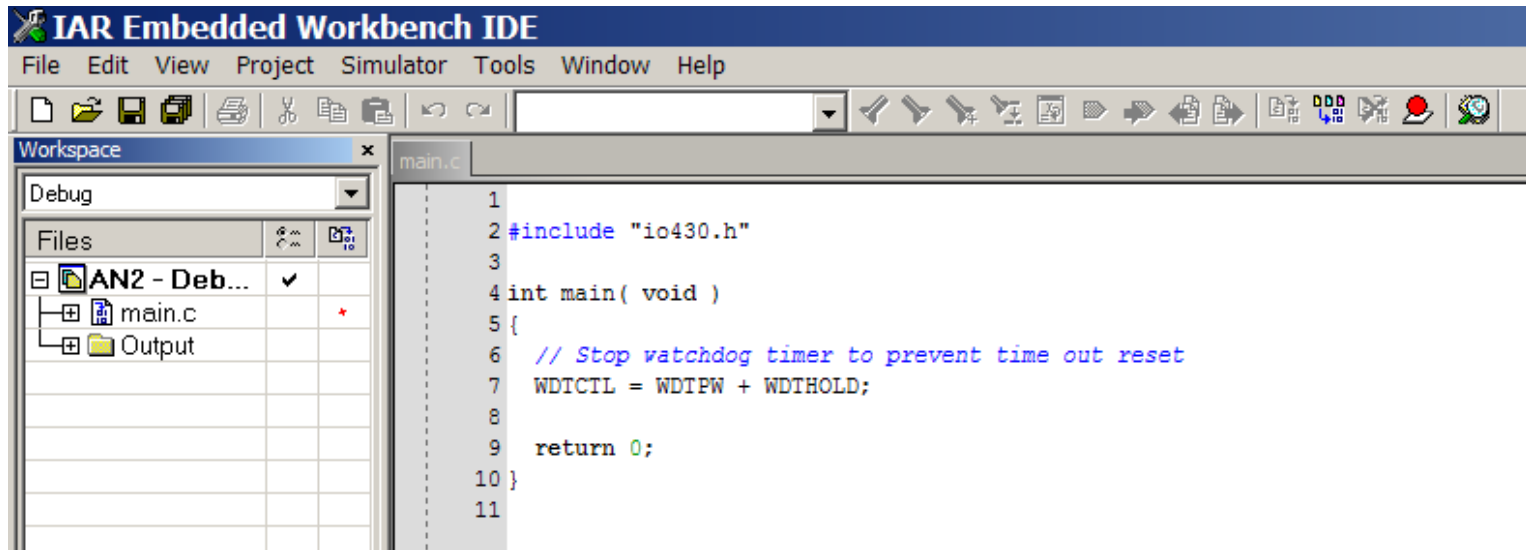
Création d'un projet C

La création d'un projet est C se déroule de la même manière qu'en assembleur. Excepté pour la fenêtre de dialogue ci-dessous



Création d'un projet C

Le fichier source principal main.c est automatiquement créé.



The screenshot shows the IAR Embedded Workbench IDE interface. The title bar reads "IAR Embedded Workbench IDE". The menu bar includes "File", "Edit", "View", "Project", "Simulator", "Tools", "Window", and "Help". The toolbar contains various icons for file operations and development. The "Workspace" panel on the left shows a project named "AN2 - Deb..." with a sub-folder "Output" and a file "main.c" marked with a red asterisk. The main editor window displays the source code for "main.c":

```
1
2 #include "io430.h"
3
4 int main( void )
5 {
6     // Stop watchdog timer to prevent time out reset
7     WDTCTL = WDTPW + WDTHOLD;
8
9     return 0;
10 }
11
```

#include io430.h

Ce fichier d'entête est très important, il sera décrit dans un autre chapitre du cours