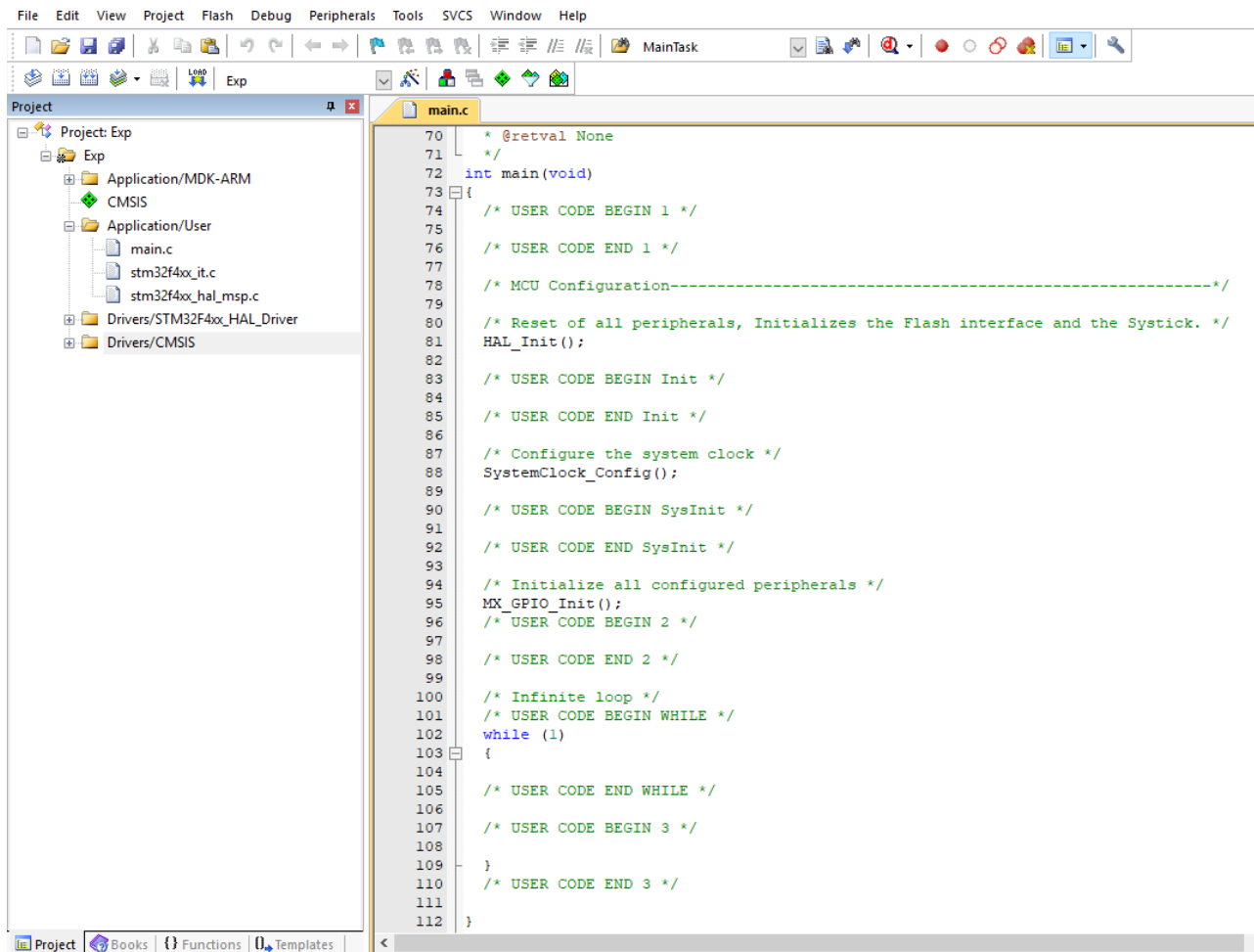


PRISE EN MAIN DU KEIL MDK-ARM

1. Keil μ vision 5 (Compilateur MDK-ARM)

1.1. L'environnement de développement Keil μ vision 5

L'environnement de développement intégré (IDE), contient deux fenêtres principales *Project Window* et la zone de travail. Dans la fenêtre *Project Window*, développez le dossier **Application/User** et ouvrez le fichier **main.c**. ».



Le fichier **main.c** contient le code source de votre programme. Vous devez insérer toujours votre code entre les balises `/* USER CODE BEGIN*/` et `/* USER CODE END*/`.

```

/* Includes -----*/
#include "main.h"
#include "stm32f4xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables -----*/
/* USER CODE BEGIN PV */
/* Private variables -----*/

/* USER CODE END PV */
/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */
    /* MCU Configuration-----*/
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */
    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */

    }
    /* USER CODE END 3 */
}

```

⇐ inclure vos librairies

⇐ déclarer vos variables et constantes.

⇐ insérer les prototypes de vos fonction.

⇐ Ecrire ici vos fonctions

⇐ Insérer ici votre code.

⇐ Insérer ici votre code.

⇐ Insérer ici votre code.

⇐ Insérer ici votre code.

⇐ Insérer ici vos instructions répétitives.

⇐ ou bien ici.

1.2. Compilation et débogage

Compilez votre programme avec la commande **Build Target** du menu **Project** ou en appuyant sur la touche F7 du clavier. A la fin de la compilation, le message suivant sera affiché :

```
Program Size: Code=2280 RO-data=440 RW-data=16 ZI-data=1024
FromELF: creating hex file...
"Exp\Exp.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:03:10
```

PRENONS COMME EXEMPLE LA SOMME DES DEUX ENTIERS SIGNES CODES SUR 16 BITS

1) Tapez la déclaration des éléments suivants dans la zone de déclaration des variables.

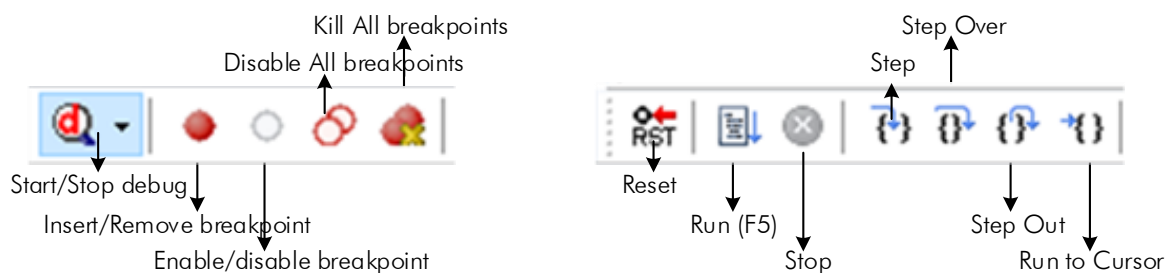
```
int16_t x1 = 0x72F1 ;
int16_t x2 = 0xF453 ;
int16_t s1 = 0x0000 ;
```

2) Dans la boucle infinie, écrire l'instruction suivante : `s1 = x1 + x2 ;`

3) Tapez F7 pur recompiler votre programme.

DEBOGAGE DU PROGRAMME

Le débogueur (Debug en Anglais) aide les développeurs à tester l'exécution de leurs programmes. Il permet de surveiller l'état des variables, des registres, ... lors du déroulement du programme. La figure suivante illustre l'ensemble des icônes utilisées par le débogueur.



- L'icône **Start/Stop debug** (Ctrl + F5) permet d'entrer ou de sortir du mode debug.
- Les points d'arrêts (Breakpoints) sont utilisés pour arrêter l'exécution à des points particuliers du programme.
- Les icônes **Reset**, **Run**, **Stop**, **Step**, **Step Over**, **Step Out** et **Run to cursor** sont utilisées pour l'exécution du programme (**Run** : lancement de l'exécution, **stop** : arrêt d'exécution, **Step** : exécution pas à pas, **Run to cursor** : exécution jusqu'à la position du curseur).

4) Appuyez sur les touches (Ctrl + F5) pour passer à l'interface de débogage.

5) Pour visualiser l'état des valeurs variables **x1**, **x2** et **s1** nous allons utiliser la fenêtre **Watch Window**. Sélectionnez la variable '**x1**' et faites un clic droit ; dans le menu qui s'ouvre, choisir la commande **Add 'x1' to watch1**. Refaire la même chose pour les autres variables.

Watch 1		
Name	Value	Type
x1	0x72F1	short
x2	0xF453	short
s1	0x0000	short
<Enter expression>		

6) Placer un point d'arrêt (breakpoint) au niveau de la ligne `s1 = x1 + x2 ;`

7) Lancez l'exécution du programme en cliquant sur l'icône **Run**.

8) Convertir en décimal les valeurs des variables **x1**, **x2** et **s1** (vous pouvez utiliser la calculatrice du PC en mode programme). Vérifiez le résultat.

9) Dans la fenêtre **Watch1**, modifier les valeurs des variables **x1 = 0x7225** et **x2 = 0x5074**. Ré-exécutez le programme et vérifiez le résultat.

2. Travail demandé

2.1. Somme des éléments d'un tableau d'entiers

Soit le tableau suivant :

Tab1	2500	15100	2000	2500
-------------	------	-------	------	------

- 1) Déclarer le tableau **Tab1** dans la zone appropriée.
- 2) Dans la boucle infinie, écrire le code de sommation des éléments du tableau **Tab1**.
- 3) Recompiler et tester votre programme.

2.2. Somme de produit des éléments de deux tableaux

Soit le deux tableaux suivants :

Tab2	200	150	20	25
-------------	-----	-----	----	----

Tab3	2	2	200	250
-------------	---	---	-----	-----

Coder et tester le programme qui permet de faire la somme de produit des éléments des deux tableaux

$$s = s + \sum_{i=0}^3 Tab2[i] * Tab3[i]$$