

BTS	MISE EN ŒUVRE Programmable System on Chip TP PSoC N° 2	TP-SN
------------	---	--------------

I. OBJECTIF DU TP

Découverte du logiciel PSoC Creator.
 Découverte de la carte de développement CY8CKIT-001.
 Mise en œuvre d'une configuration matérielle, programmation logicielle.

II. BUT DU TP

Le traditionnel « Hello Word » à afficher sur l'écran LCD de la carte de développement.
 Puis utilisation de compteurs pour réaliser un chronomètre.
 Les apprentissages seront réutilisés pour les TP suivants ainsi que pour le mini projet.

III. CRÉATION D'UN PROJET SOUS PSoC CREATOR

Lancer PSoC Creator à l'aire de son icône :



Créer un nouveau projet en utilisant la commande :
 Cliquer sur **File** puis sur **New** puis sur **Project**.

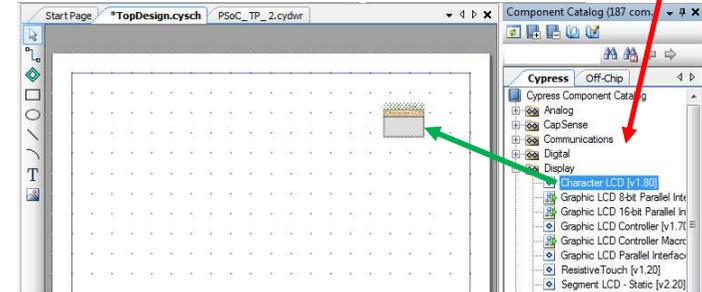
Sélectionner **Empty PSoC 3 Design** (conception d'un projet vide) puis nommer le projet et indiquer le répertoire parent du projet, faire ensuite OK.



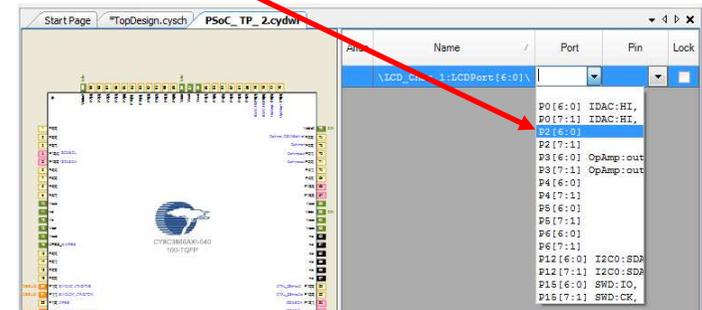
IV. MISE EN ŒUVRE SIMPLE D'UN AFFICHEUR LCD

IV.1. CONFIGURATION MATÉRIELLE

Placer sur la feuille de schéma **PSoC_TP_2.cysch** le composant **Character LCD** (contrôleur d'afficheur LCD alphanumérique) à partir de la librairie des composants dans la partie **Display** en cliquant dessus puis en effectuant un glisser vers la fenêtre de schéma. Renommer celui-ci en LCD.



Attribuer les broches P2.0 à P2.6 à ce composant en ouvrant la fenêtre **PSoC_TP_2.cydwr** :



IV.2. RÉALISATION DU TRAITEMENT LOGICIEL

Consulter la documentation du contrôleur d'afficheur LCD (bouton Datasheet après double-clic sur le composant). Un grand nombre de fonctions sont disponibles :

Functions	Description
LCD_Char_Start()	Starts the module and loads custom character set to LCD, if it was defined.
LCD_Char_Stop()	Turns off the LCD
LCD_Char_DisplayOn()	Turns on the LCD module's display
LCD_Char_DisplayOff()	Turns off the LCD module's display
LCD_Char_PrintString()	Prints a null-terminated string to the screen, character by character
LCD_Char_PutChar()	Sends a single character to the LCD module data register at the current position.
LCD_Char_Position()	Sets the cursor's position to match the row and column supplied

Functions	Description
LCD_Char_Init()	Performs initialization required for component's normal work
LCD_Char_Enable()	Turns on the display
LCD_Char_DisplayOn()	Turns on the LCD module's display
LCD_Char_DisplayOff()	Turns off the LCD module's display
LCD_Char_PrintString()	Prints a null-terminated string to the screen, character by character
LCD_Char_PutChar()	Sends a single character to the LCD module data register at the current position.
LCD_Char_Position()	Sets the cursor's position to match the row and column supplied
LCD_Char_PrintInt8()	Prints a two-ASCII-character hex representation of the 8-bit value to the Character LCD module.
LCD_Char_PrintInt16()	Prints a four-ASCII-character hex representation of the 16-bit value to the Character LCD module.
LCD_Char_PrintNumber()	Prints the decimal value of a 16-bit value as left-justified ASCII characters

Ouvrir le fichier main.c et le compléter comme ci-dessous :

```

Start Page TopDesign.cysch main.c
1  /* ----- */
2  * Fichier main.c du projet PSoC_TP_2
3  * ----- */
4  /*
5  #include <device.h>
6
7  void main()
8  {
9      /* Place your initialization/startup code here (e.g. MyInst_Start()) */
10     LCD_Init();           //Initialise le dialogue avec l'afficheur LCD
11     LCD_Start();         //D marre le dialogue avec l'afficheur LCD
12     LCD_DisplayOn();     //Allume l'afficheur LCD
13     /* CyGlobalIntEnable; */ /* Uncomment this line to enable global interrupts. */
14     LCD_PrintString("Hello Word"); //Affiche une cha ne de caract res
15     LCD_Position(1,0);   //Positionne l'affichage ligne 1 colonne 0
16     LCD_PrintString("#Bonjour a tous#"); //Affiche une cha ne de caract res
17     for(;;)              //Boucle sans fin
18     {                    //D but de la boucle sans fin
19         /* Place your application code here. */
20     }                    //Fin de la boucle sans fin
21 }
22 /* [] END OF FILE */

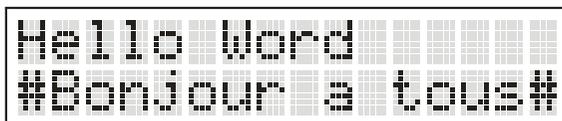
```

IV.3. CONSTRUCTION DU PROJET ET PROGRAMMATION

Il ne reste plus qu'à construire le projet, corriger les  ventuelles erreurs puis programmer le composant.

IV.4. ESSAI

L' cran de l'afficheur LCD doit pr senter l'image donn e ci-dessous :

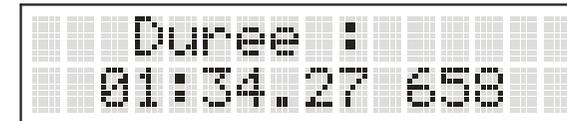


V. MISE EN ŒUVRE D'UN PREMIER CHRONOMÈTRE

On d sire afficher la dur e pendant laquelle le bouton-poussoir SW1 est actionn . Si l'on appuie plusieurs fois les dur es doivent s'ajouter.

Un appui sur le bouton-poussoir SW2 pendant que le bouton-poussoir SW1 est rel ch  provoque la remise   z ro du comptage.

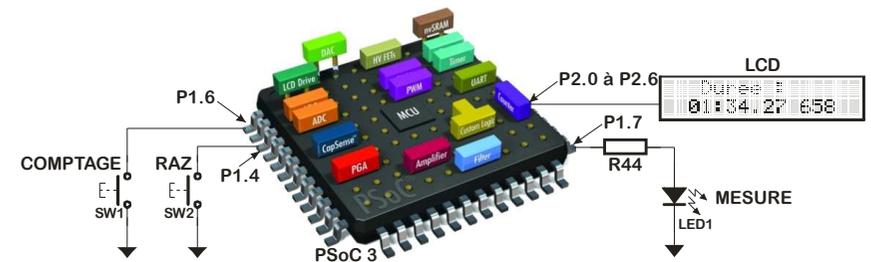
L'affichage doit  tre conforme   l'image donn e ci-dessous (Heure : minute . seconde milli me de seconde) :



Ce premier chronom tre sera r alis  avec une architecture assez semblable   ce qui serait fait avec un microcontr leur : un compteur/timer (sur 16 ou 32 bits) permettant de mesurer la dur e d'activation du bouton-poussoir. Les calculs pour r cup rer les secondes, minutes et heures ainsi que leur affichage  tant r alis s logiciellement.

V.1. SCH MA STRUCTUREL EXT RIEUR

Voici le sch ma structurel ext rieur. Les composants sont implant s sur la carte de d veloppement.



V.2. CONFIGURATION MAT RIELLE

Ajouter   la page de sch ma deux broches d'entr es num riques (**Digital Input Pin**) qui seront renomm es COMPTAGE et RAZ.

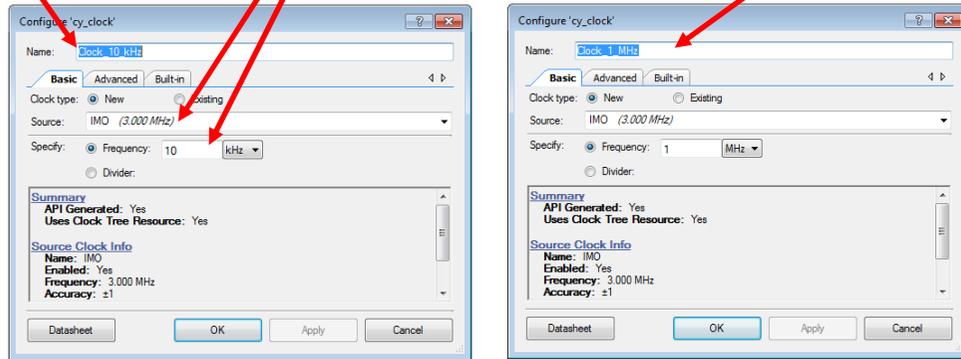
Configuration : **Drive Mode : Resistive Pull Up** et **Initial State : 1**.

Ajouter une broche de sortie num rique (**Digital Output Pin**) qui sera renomm e MESURE. D cocher la case **HW Connection**.

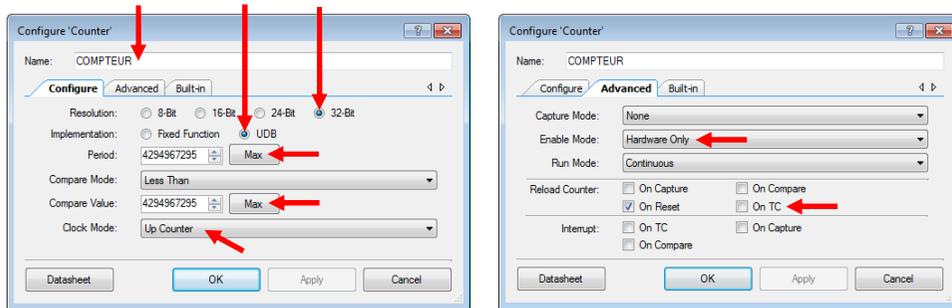
Affectation des broches : P1.6 pour COMPTAGE, P1.4 pour RAZ et P1.7 pour MESURE.

Ajouter deux inverseurs (**Not** dans la partie **Digital** puis sous-partie **Logic**) et un et (**And** dans la m me partie).

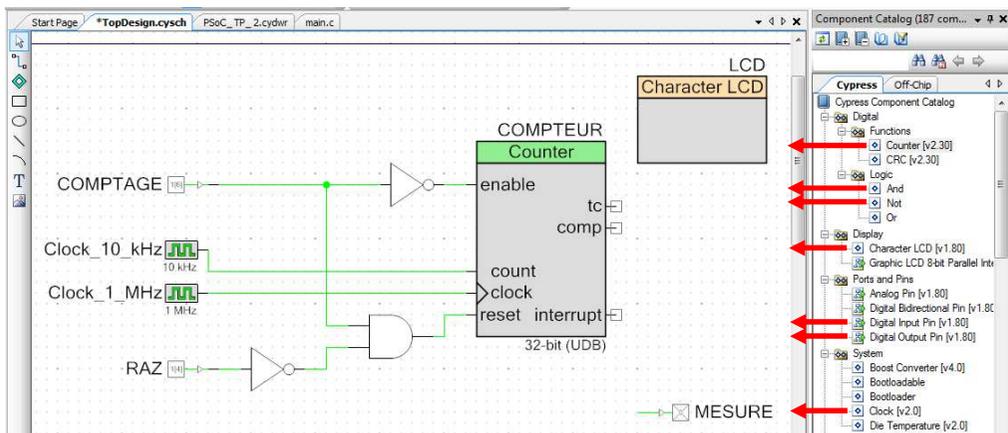
Ajouter deux horloges (**Clock** dans la partie **System**) qui seront renommées **Clock_1_MHz** et **Clock_10_kHz** et les configurer comme ci-dessous :



Ajouter un compteur (**Counter** dans la partie **Digital** puis sous-partie **Functions**). Celui-ci sera configuré afin de fonctionner en compteur, d'avoir une entrée de validation, une entrée de comptage, et une taille de 32 bits. Consulter la documentation technique pour information.



Disposer les composants et effectuer les liaisons (**Wire Tools**) comme ci-dessous :



V.3. PREMIÈRE PARTIE DU TRAITEMENT LOGICIEL

Editer le fichier **main.c**. Le compléter comme ci-dessous :

```

1  /* =====
2  * Fichier main.c du projet PSoC_TP_2
3  * =====
4  */
5  #include <device.h>
6
7  void main()
8  {
9      uint32 VALEUR = 0; //Nombre dans lequel sera copié la valeur du compteur
10     LCD_Init(); //Initialise le dialogue avec l'afficheur LCD
11     LCD_Start(); //Démarre le dialogue avec l'afficheur LCD
12     LCD_DisplayOn(); //Allume l'afficheur LCD
13     COMPTEUR_Start(); //Configure le compteur et valide son fonctionnement
14     for(;;) //Boucle sans fin
15     { //Début de la boucle sans fin
16         VALEUR = COMPTEUR_ReadCounter(); //Sauvegarde la valeur courante du compteur
17         LCD_Position(0,6); //Positionne l'affichage ligne 0 colonne 6
18         LCD_PrintInt16(VALEUR); //Affiche valeur en hexadécimal sur le LCD
19     } //Fin de la boucle sans fin
20 }
21 /* [] END OF FILE */
    
```

V.4. CONSTRUCTION DU PROJET, PROGRAMMATION ET ESSAI

Construire le projet, programmer le composant puis effectuer le premier essai.

L'afficheur doit indiquer la valeur du compteur en hexadécimal. Cette valeur est : 0000. Appuyer sur le bouton-poussoir **COMPTAGE** (SW1) pendant quelques instants. Le nombre affiché doit s'incrémenter très rapidement et se figer quand le bouton-poussoir est relâché. Un appui sur le bouton-poussoir **RAZ** (SW2) permet de remettre le compteur à 0000.

Observer le signal **MESURE** (P1.7) à l'oscilloscope. Mesurer le temps mis pour traiter l'affichage de la valeur du compteur (durée à l'état haut). Quelle est la fréquence maximale d'affichage de cette valeur ? 2 kHz

V.5. SECONDE PARTIE DU TRAITEMENT LOGICIEL

On désire maintenant afficher le temps comme prévu. Il est donc nécessaire d'effectuer des calculs afin d'afficher les millièmes de seconde, les secondes, les minutes et les heures. Il suffit d'utiliser la division entière (/) et le reste de la division entière (% ou modulo).

La fonction **sprintf()**, standard du langage C, déclarée dans le fichier d'entête **stdio.h** permet de réaliser l'affichage de plusieurs variables de types différents. Le formatage **%02u** permet d'afficher un nombre entier non signé (sur 16 bits) sur 2 caractères avec des 0 nos significatifs devant. Le (uint16) devant les variables effectue une promotion.

Modifier le fichier main.c comme ci-dessous :

```
Start Page TopDesign.cysch main.c
1 /* =====
2  * Fichier main.c du projet PSoC_TP_2
3  * =====
4 */
5 #include <device.h>
6 #include <stdio.h>
7
8 void main()
9 {
10     uint32 VALEUR = 0;           //Nombre dans lequel sera copié la valeur du compteur
11     uint16 MILLIEMES;          //Nombre contenant le résultat du calcul des millièmes
12     uint8 SECONDES;            //Nombre contenant le résultat du calcul des secondes
13     uint8 MINUTES;             //Nombre contenant le résultat du calcul des Minutes
14     uint8 HEURES;              //Nombre contenant le résultat du calcul des heures
15     unsigned char LIGNE[13];   //Réserve une chaîne de 13 car. pour l'affichage
16
17     LCD_Init();                //Initialise le dialogue avec l'afficheur LCD
18     LCD_Start();               //Démarre le dialogue avec l'afficheur LCD
19     LCD_DisplayOn();           //Allume l'afficheur LCD
20     COMPTEUR_Start();          //Configure le compteur et valide son fonctionnement
21     LCD_Position(0,3);         //Positionne l'affichage ligne 0 colonne 3
22     LCD_PrintString("Duree :"); //Affiche une chaîne de caractères
23
24     for(;;)                    //Boucle sans fin
25     {                          //Début de la boucle sans fin
26         VALEUR = COMPTEUR_ReadCounter(); //Sauvegarde la valeur courante du compteur
27         MESURE_Write(1);         //Permet de mesurer le temps du traitement
28         VALEUR += 5;             //Prépare à l'arrondi
29         VALEUR /= 10;           //VALEUR est arrondi au millième de seconde
30         MILLIEMES = VALEUR % 1000; //Calcule le nombre de millièmes de secondes
31         VALEUR /= 1000;         //VALEUR est arrondie à la seconde
32         SECONDES = VALEUR % 60; //Calcule le nombre de secondes
33         VALEUR /= 60;           //VALEUR est arrondie à la seconde
34         MINUTES = VALEUR % 60; //Calcule le nombre de minutes
35         VALEUR /= 60;           //VALEUR est arondie à l'heure
36         HEURES = VALEUR;        //Calcule le nombre d'heures
37         LCD_Position(1,2);      //Positionne l'affichage ligne 1 colonne 2
38         sprintf(LIGNE, "%02u:%02u.%02u %03u", (uint16)HEURES, (uint16)MINUTES,
39                 (uint16)SECONDES, MILLIEMES);
40         //Construit la ligne à afficher
41         LCD_PrintString(LIGNE); //Affiche la seconde ligne sur l'afficheur
42         MESURE_Write(0);        //Permet de mesurer le temps de l'affichage
43     }                          //Fin de la boucle sans fin
44 }
45 /* [] END OF FILE */
```

V.6. CONSTRUCTION DU PROJET, PROGRAMMATION ET ESSAI

Reconstruire le projet, programmer le composant puis effectuer un nouvel essai.

L'afficheur doit respecter le Cahier des Charges initial.
Appuyer sur le bouton-poussoir COMPTAGE (SW1) pendant quelques instants. L'affichage doit évoluer au cours du temps jusqu'à se figer quand le bouton-poussoir est relâché.
Un appui sur le bouton-poussoir RAZ (SW2) permet de remettre le compteur à 0.

Effectuer une mesure de la durée d'activation de COMPTAGE avec un oscilloscope et comparer les valeurs trouvées.

Effectuer la mesure du temps du traitement (calculs + affichage). 2.2 ms

Des 2 traitements, quel est le plus gourmand en temps machine ?

L'affichage est-il capable d'afficher les millisecondes ?

Avec un compteur 32 bits, quel est la durée maximale mesurable sachant que le compteur s'incrémente toutes les 100 µs (10 kHz) ? 4 j 23 h 18 min 16 s 730 mil

VI. MISE EN ŒUVRE D'UN SECOND CHRONOMÈTRE

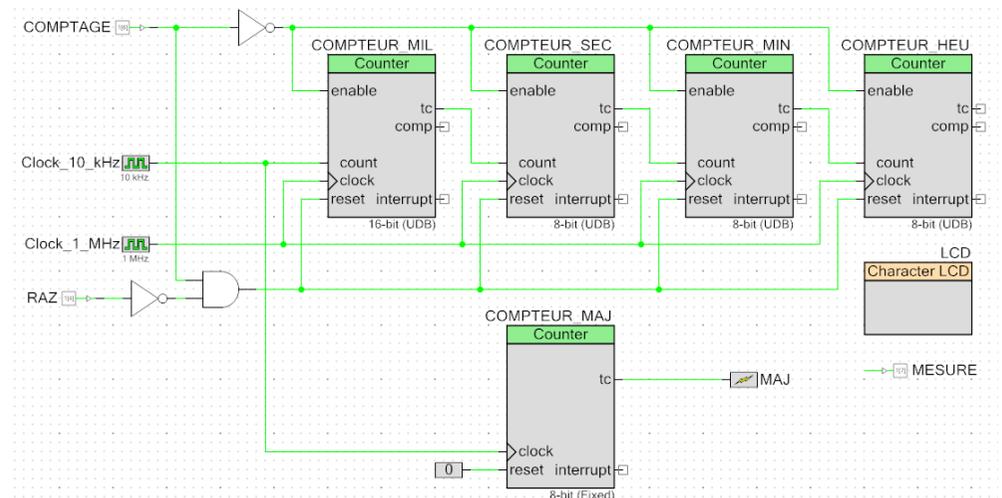
On désire simplifier la partie programme. Il faut donc reporter une partie du traitement logiciel dans le traitement matériel.

Une des solutions est d'utiliser plusieurs compteurs qui seront cascades afin d'obtenir les millièmes, secondes, minutes et heures dans des variables distinctes et ainsi éviter les calculs.

Pour l'affichage, un traitement moins répétitif permettra de disposer de temps machine pour un autre traitement par exemple. Il est inutile d'essayer d'afficher en temps réel les millisecondes sur l'afficheur LCD car l'œil ne peut suivre la cadence. In affichage toutes les 20 ms environ est amplement suffisant.

VI.1. CONFIGURATION MATÉRIELLE

Placer les nouveaux composants suivant la disposition suivante :

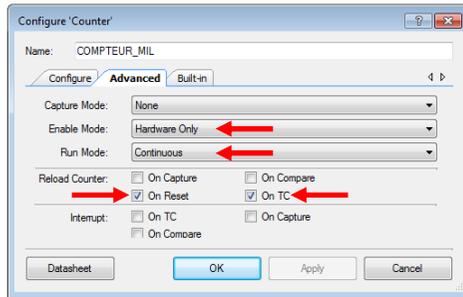
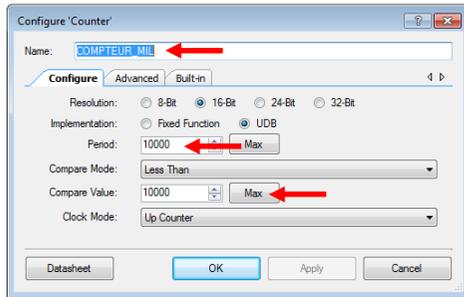


Afin de pouvoir disposer de toutes les broches sur les compteurs, modifier leurs noms et leurs configurations comme ci-après :

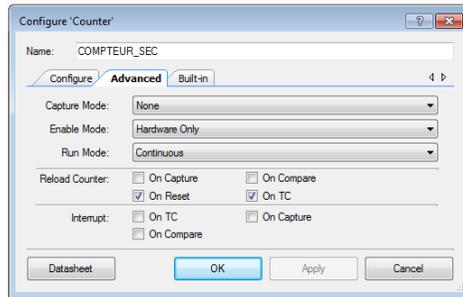
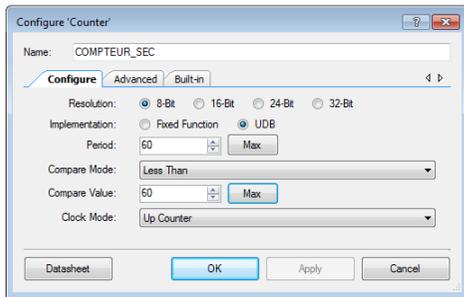
Le premier compteur **COMPTEUR_MIL** effectue le comptage des dixièmes de millisecondes (pour plus de précision) sur 16 bits (**Résolution : 16 bits, Implémentation : UDB**). C'est un diviseur par 10.000 (**Période : 10000**). La fréquence du signal sur l'entrée count étant de 10 kHz on obtient bien sur la sortie tc une fréquence de 1 Hz, soit une période de 1 s.

Le comptage s'effectue seulement lorsque l'entrée enable est validée (**Enable Mode : Hardware Only**).

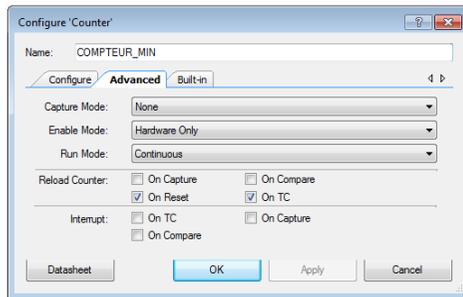
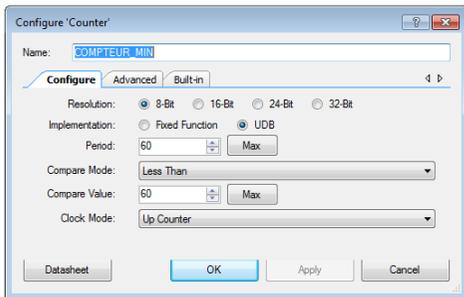
Il fonctionne en continu (**Run Mode : Continuous**) et est rechargé à 0 lorsque qu'il est arrivé en fin de comptage (**On TC**) ou lorsque l'entrée reset est activée (**On Reset**).



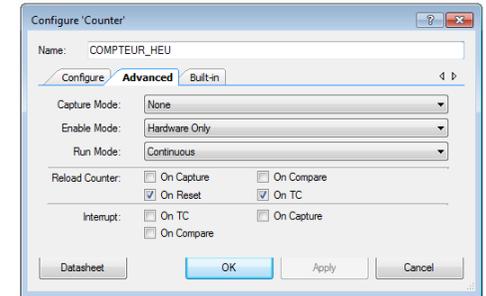
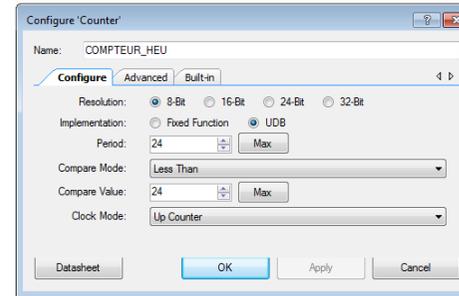
Le compteur **COMPTEUR_SEC** effectue le comptage des secondes. C'est un diviseur par 60.



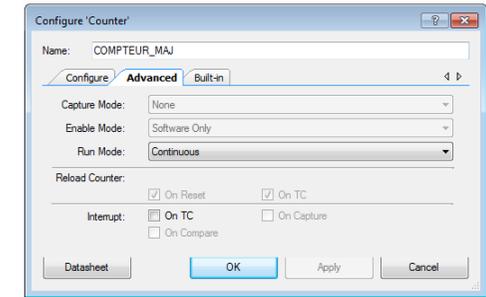
Le compteur **COMPTEUR_MIN** effectue le comptage des minutes. C'est un diviseur par 60.



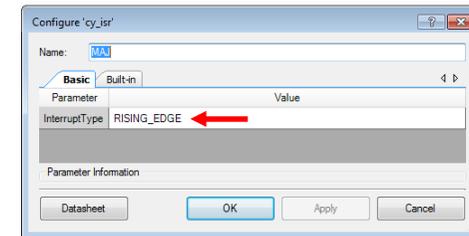
Le compteur **COMPTEUR_HEU** effectue le comptage des heures. C'est un diviseur par 24.



Le compteur **COMPTEUR_MAJ** est utilisé pour générer une interruption afin de mettre à jour l'affichage. La période est de 211 coups d'horloge, soit 21,1 ms ce qui donne 47 Hz. 211 est un nombre premier afin d'avoir un défilement plausible des millisecondes.



L'interruption connectée à la sortie tc de **COMPTEUR_MAJ** doit être configuré comme ci-dessous :



Terminer par les connexions entre les composants.

VI.2. TRAITEMENT LOGICIEL

Editer le fichier main.c et le compléter comme ci-après :

```

1  /*
2  * Fichier main.c du projet PSoC_TP_2
3  *
4  */
5  #include <device.h>
6
7  uint8 MISE_A_JOUR;           //Déclaration d'une variable globale (flag)
8                               //pour dialogue entre la fonction main.c
9                               //et la fonction l'interruption CY_ISR(MAJ_Interrupt)
10
11 void main()
12 {
13     uint16 MILLIEMES;        //Nombre contenant le résultat du calcul des millièmes
14     uint8 SECONDES;          //Nombre contenant le résultat du calcul des secondes
15     uint8 MINUTES;          //Nombre contenant le résultat du calcul des Minutes
16     uint8 HEURES;           //Nombre contenant le résultat du calcul des heures
17     uint8 CARACTERE;        //Caractère utilisé pour transmettre au LCD
18     LCD_Init();             //Initialise le dialogue avec l'afficheur LCD
19     LCD_Start();            //Démarre le dialogue avec l'afficheur LCD
20     LCD_DisplayOn();        //Allume l'afficheur LCD
21     LCD_PrintString(" Duree :"); //Affiche une chaîne de caractères
22     COMPTEUR_MIL_Start();    //Configure le compteur et valide son fonctionnement
23     COMPTEUR_SEC_Start();    //Configure le compteur et valide son fonctionnement
24     COMPTEUR_MIN_Start();    //Configure le compteur et valide son fonctionnement
25     COMPTEUR_HEU_Start();    //Configure le compteur et valide son fonctionnement
26     COMPTEUR_MAJ_Start();    //Configure et valide l'interruption MAJ
27     MAJ_Start();            //Configure et valide l'interruption correspondante
28     CyGlobalIntEnable;      //Valide les interruptions globalement
29     for(;;)                 //Boucle sans fin
30     {
31         //Début de la boucle sans fin
32         if (MISE_A_JOUR)     //Si demande de mise jour de l'affichage
33         {
34             MISE_A_JOUR = 0; //Efface la demande de mise à jour
35             HEURES = COMPTEUR_HEU_ReadCounter(); //Sauvegarde dans les variables
36             MINUTES = COMPTEUR_MIN_ReadCounter(); //correspondantes les valeurs
37             SECONDES = COMPTEUR_SEC_ReadCounter(); //courantes des quatre compteurs
38             MILLIEMES = COMPTEUR_MIL_ReadCounter(); //
39             MESURE_Write(1); //Permet de mesurer le temps du traitement
40             LCD_Position(1,2); //Positionne l'affichage ligne 1 colonne 2
41             CARACTERE = HEURES / 10; //Extrait le chiffre des dizaines des heures
42             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
43             CARACTERE = HEURES % 10; //Extrait le chiffre des unités des heures
44             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
45             LCD_WriteData(':'); //Envoi le caractère : au LCD
46             CARACTERE = MINUTES / 10; //Extrait le chiffre des dizaines des minutes
47             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
48             CARACTERE = MINUTES % 10; //Extrait le chiffre des unités des minutes
49             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
50             LCD_WriteData('.'); //Envoi le caractère . au LCD
51             CARACTERE = SECONDES / 10; //Extrait le chiffre des dizaines des secondes
52             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
53             CARACTERE = SECONDES % 10; //Extrait le chiffre les unités des secondes
54             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
55             LCD_WriteData(' '); //Envoi le caractère espace au LCD
56             MILLIEMES = (MILLIEMES + 5) / 10; //Calcule l'arrondi au millième de seconde
57             CARACTERE = MILLIEMES / 100; //Extrait le chiffre des centaines des millièmes
58             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
59             MILLIEMES = MILLIEMES % 100; //Supprime les centaines des millièmes
60             CARACTERE = MILLIEMES / 10; //Extrait le chiffre des dizaines des millièmes
61             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
62             CARACTERE = MILLIEMES % 10; //Extrait le chiffre des unités des millièmes
63             LCD_WriteData(CARACTERE + 48); //Envoi le caractère correspondant au LCD
64             MESURE_Write(0); //Permet de mesurer le temps du traitement.
65         }
66     }
67 }
68 /* END OF FILE */

```

Compléter le fichier MAJ.c (dans **Generated_Source** puis **PSoC3**) comme ci-dessous :
Ligne 28 :

```

24  /*
25  * Place your includes, defines and code here
26  *
27  * #START MAJ_intc */
28  extern uint8 MISE_A_JOUR;
29  * #END */

```

Ligne 133 :

```

128  /*
129  * #START MAJ_Interrupt)
130  *
131  * Place your Interrupt code here. */
132  * #START MAJ_Interrupt */
133  MISE_A_JOUR = 1;
134  * #END */

```

Plusieurs compteurs seront mis en œuvre, d'ont l'un servira à mettre à 1 le drapeau (flag) `Mise_a_Jour` toutes les 21ms par interruption. L'affichage se fera lorsque le drapeau sera à un et celui-ci sera remis à 0.

VI.3. CONSTRUCTION DU PROJET, PROGRAMMATION ET ESSAI

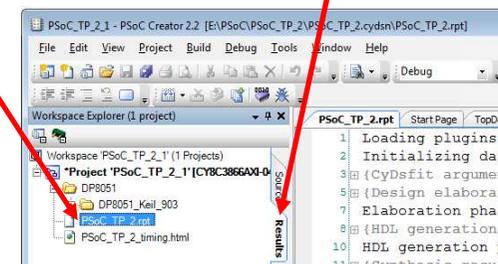
Reconstruire le projet, programmer le composant puis effectuer un nouvel essai.

Vérifier que le fonctionnement est bien celui attendu.

Observer à l'oscilloscope le signal `MESURE` (P1.7). Mesurer la durée au niveau haut et bas et commenter ses deux états.

VII. ANALYSE DES FICHIERS DE RAPPORTS

Editer le fichier de rapport : cliquer sur l'onglet `Results` puis double-cliquer sur le fichier `PSoC_TP_2.rpt`.



```

Start Page TopDesign.cysch PSoc_TP_2_1.cydwr main.c PSoc_TP_2_1.rpt
1690 -----
1691 Technology mapping summary
1692 -----
1693
1694 Resource Type           : Used : Free : Max : % Used
1695 =====
1696 Digital domain clock dividers :    2 :    6 :    8 : 25.00%
1697 Analog domain clock dividers :    0 :    4 :    4 :  0.00%
1698 Pins                     :   13 :   59 :   72 : 18.06%
1699 UDB Macrocells           :   30 :  162 :  192 : 15.63%
1700 UDB Unique Pterms        :   25 :  359 :  384 :  6.51%
1701 UDB Total Pterms         :   33 :    :    :    :
1702 UDB Datapath Cells       :    5 :   19 :   24 : 20.83%
1703 UDB Status Cells        :    4 :   20 :   24 : 16.67%
1704     StatusI Registers    :    4 :    :    :    :
1705 UDB Control Cells        :    0 :   24 :   24 :  0.00%
1706 DMA Channels             :    0 :   24 :   24 :  0.00%
1707 Interrupts               :    1 :   31 :   32 :  3.13%
1708 DSM Fixed Blocks         :    0 :    1 :    1 :  0.00%
1709 VIDAC Fixed Blocks       :    0 :    4 :    4 :  0.00%
1710 SC Fixed Blocks          :    0 :    4 :    4 :  0.00%
1711 Comparator Fixed Blocks   :    0 :    4 :    4 :  0.00%
1712 Opamp Fixed Blocks       :    0 :    4 :    4 :  0.00%
1713 CapSense Buffers         :    0 :    2 :    2 :  0.00%
1714 CAN Fixed Blocks         :    0 :    1 :    1 :  0.00%
1715 Decimator Fixed Blocks    :    0 :    1 :    1 :  0.00%
1716 I2C Fixed Blocks         :    0 :    1 :    1 :  0.00%
1717 Timer Fixed Blocks       :    1 :    3 :    4 : 25.00%
1718 DFB Fixed Blocks         :    0 :    1 :    1 :  0.00%
1719 USB Fixed Blocks         :    0 :    1 :    1 :  0.00%
1720 LCD Fixed Blocks         :    0 :    1 :    1 :  0.00%
1721 EMIF Fixed Blocks        :    0 :    1 :    1 :  0.00%
1722 LPF Fixed Blocks         :    0 :    2 :    2 :  0.00%
1723 -----
1764 PLD Packing Summary
1765 -----
1766
1767 Resource Type : Used : Free : Max : % Used
1768 =====
1769 PLDs : 12 : 36 : 48 : 25.00%

```

Au vu du fichier de rapport, pourrait-on ajouter un second chronomètre identique au premier ?

En utilisant un afficheur LCD de 4 lignes de 20 caractères, pourrait-on implémenter 4 chronomètres dans ce composant ? Quels seraient les composants communs aux 4 chronomètres ?

Pour les calculs et le transfert à l'affichage, quelle solution pourrait être employée ?

Si l'on voulait utiliser le bouton-poussoir COMPTAGE comme sur un chronomètre classique, à savoir un appui pour lancer le comptage et un autre pour l'arrêt, quel composant pourrait être ajouté ?

Consulter le fichier PSoc_TP_2_Timing.html