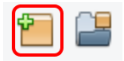
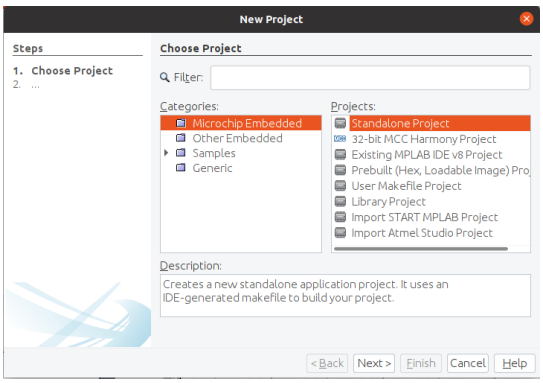
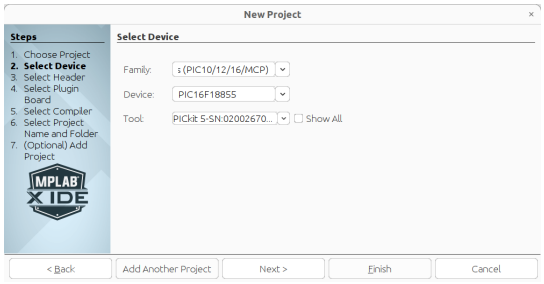
2 Créer un projet MPLABX

Lors du tutoriel en langage C, vous aviez utilisé un projet existant. Ici, au contraire, vous allez apprendre à en créer un, puis à le configurer avec les ressources matérielles nécessaires à votre application. Pour chaque nouveau sujet d'APP, il faudra commencer par cette étape.

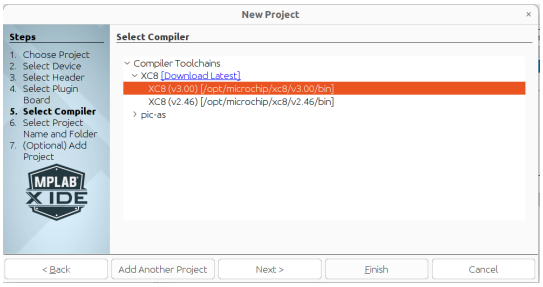
Dans le bandeau des icônes de raccourcis, **cliquer** sur l'icône `New Project`  , celle de gauche.



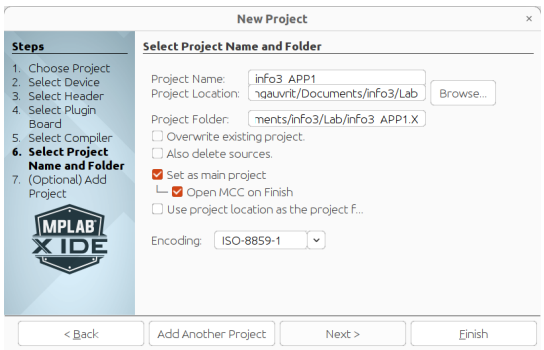
(a) Choisir le type de projet.



(b) Choisir le PIC, le 16F18855 ici.



(c) Choisir la version du compilateur.



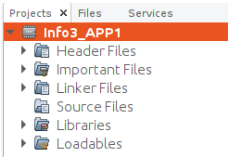
(d) Choisir le nom et le dossier du projet.

FIGURE 2.1 – Fenêtres pour créer un projet.

La création d'un projet consiste à sélectionner dans l'ordre :

1. le type du projet comme sur la figure 2.1a ;
2. le microntrôleur PIC, ici le PIC16F18855 comme sur la figure 2.1b ;
3. la version du compilateur, choisir la version la plus récente si vous en avez plusieurs de proposer comme sur la figure 2.1c ;

4. le nom de votre projet (`Info3_APP1`) et son emplacement de stockage comme sur la figure 2.1d. Nous vous conseillons de le stocker dans votre dossier `Info3` sur le Bureau (en local sur la machine).



Terminer en cliquant sur `Finish` . Le projet se crée et s'ouvre dans l'onglet `Project` . Pour l'instant, il n'y a pas de fichier `main.c` créé dans le dossier `Source Files` . C'est à la fin de la phase de configuration qu'il le sera.

Remarque

À noter que l'outil `Microchip Code Configurator` (MCC) se lance automatiquement à la création du projet.

3 Configurer son projet avec MCC

3.1 Installer et Démarrer MCC

On configure notre projet grâce à un outil très puissant, `Microchip Code Configurator` (MCC). **Cliquer** sur l'icône (si MCC n'a pas été lancé lors de la création du projet).

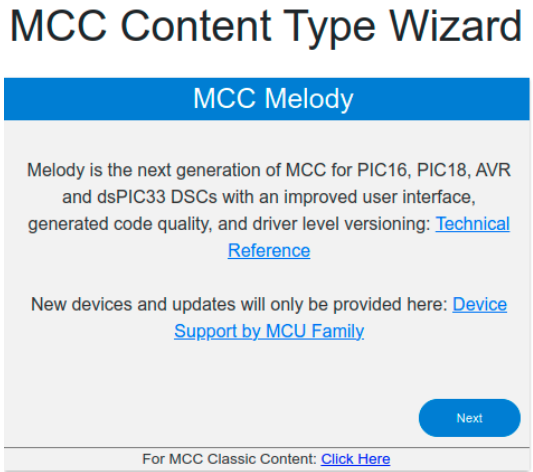


FIGURE 3.1 – Fenêtre `Content Manager` à l'ouverture de MCC.

Une fenêtre `Content Manager` s'ouvre. **Cliquer** sur `Next`. L'onglet `MCC` apparaît avec les outils pour la configuration du projet.

3.2 Configurer son projet avec MCC

3.2.1 Configurer les entrées-sorties numériques du PIC

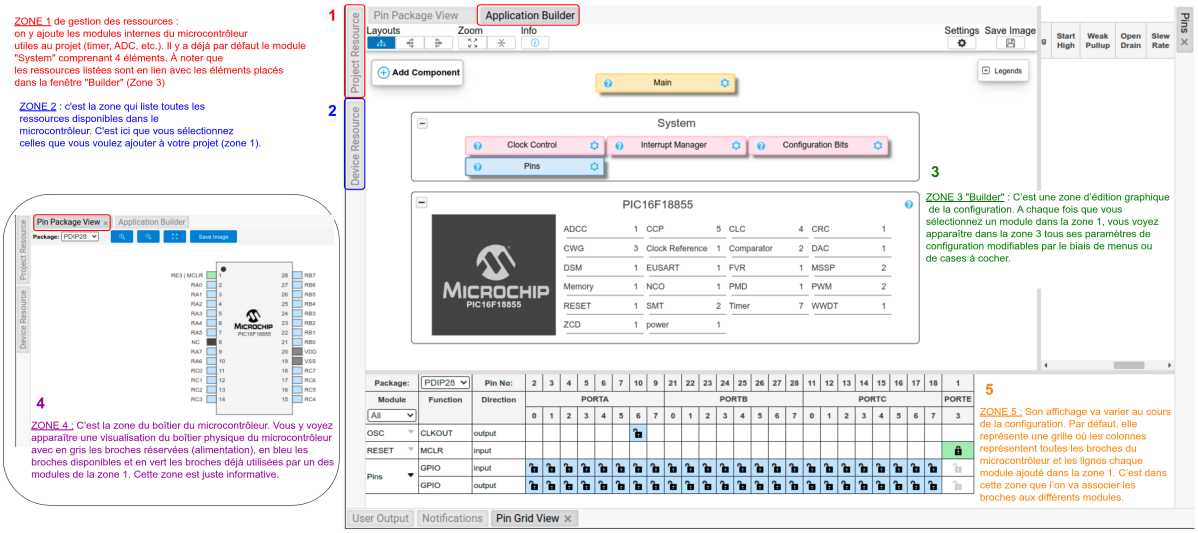
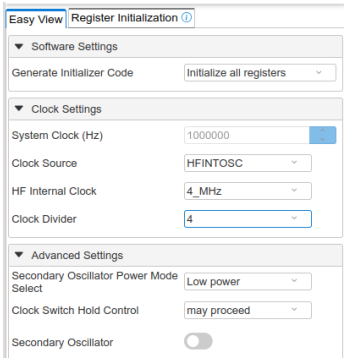


FIGURE 3.2 – Ouverture de MCC Melody.

Changer le type de boîtier Au préalable, nous allons changer le type de boîtier du circuit. En effet, dans le champ `Package` de la zone 4, le boîtier est de type `SOIC28`, **choisir** `PDIP28` dans le menu.



Bien repérer les différentes zones sur la figure 3.2. Le principe de la configuration consiste à sélectionner parmi les ressources du PIC16F18855 (zone 2 : Device Resource) celles nécessaires pour notre projet (zone 1 : Project Resource). À noter que les modules choisis pour votre projet apparaissent automatiquement dans la zone 3 : Builder. Ainsi le module `Clock Control` est déjà présent dans la zone 1 et donc dans la zone 3. Changeons sa configuration. **Cliquer** sur ce module. Une fenêtre de configuration s'ouvre, on modifie les paramètres `HF Internal Clock` et `Clock Divider` comme sur la figure ci-contre.
On règle la fréquence de l'oscillateur intégré dans le PIC sur 4 Mhz. Elle est divisée par 4 pour fournir un signal d'horloge à 1 MHz qui cadence le microcontrôleur. On est bien loin des fréquences de plusieurs GHz des processeurs de nos ordinateurs ! Dans un microcontrôleur, nous n'avons pas besoin de telles cadences ! On veut surtout un circuit économe en énergie, dégageant peu de chaleur.

Configurer les sorties numériques Le module `Pins` est aussi présent dans la zone 1 et dans la zone 3 ainsi que dans l'interface graphique `Pin Grid View` de la zone 5. Nous

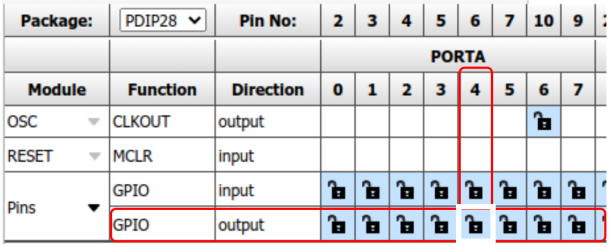
pouvons alors configurer les broches d'entrées-sorties numériques. À partir de la zone 3, **cliquer** sur `Pins` à l'intérieur du module `System` (on peut aussi le faire à partir de la zone 1). La fenêtre MCC bascule sur l'onglet `Pin Package View` et une fenêtre `Pins` s'ouvre à droite. Nous la complèterons dans un deuxième temps.
Nous allons commencer par configurer les LEDs présentes aux 4 coins de la carte PIC. **Consulter** le schéma de la carte d'extension (dans les ressources) et notez dans le tableau ci-dessous les ports (de la forme RXY où X = lettre du port et Y = numéro de la ligne du port X) sur lesquels sont connectées les LEDs (cela vous permettra de rapidement configurer les projets ultérieurs) :

Nom de la broche	Port d'entrée-sortie du PIC	entrée ou sortie
BP	R__	
LED_STOP	R__	
LED_GAUCHE	R__	
LED_DROIT	R__	

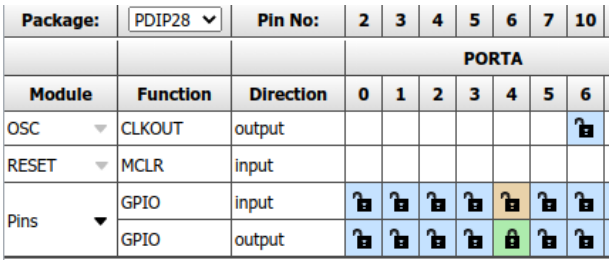
TABLE 3.1 – Ports d'entrées-sorties numériques du PIC.

Comme nous voulons utiliser la `LED STOP` sur la broche `RA4`, il va falloir la configurer en sortie. Rien de plus simple dans le CODE CONFIGURATOR :

— Dans la zone 5 (`Pins Grid view`), repérer la colonne `RA4` et la ligne `GPIO Output`.
— À l'intersection, vous voyez une icône en forme de cadenas ouvert sur un fond bleu :



— **Cliquer** sur le cadenas pour le verrouiller. L'icône se transforme en cadenas fermé sur un fond vert. Cela signifie que l'on a attribué à RA4 le comportement d'une broche numérique en sortie et que celle-ci ne pourra plus être utilisée pour autre chose.



On peut également remarquer une icône d'une autre couleur apparue sur la ligne 'input'. Ce cadenas ouvert avec un fond couleur pêche indique que la broche est déjà utilisée.

Refaire les mêmes opérations pour les deux autres LEDs. Vous devriez obtenir :

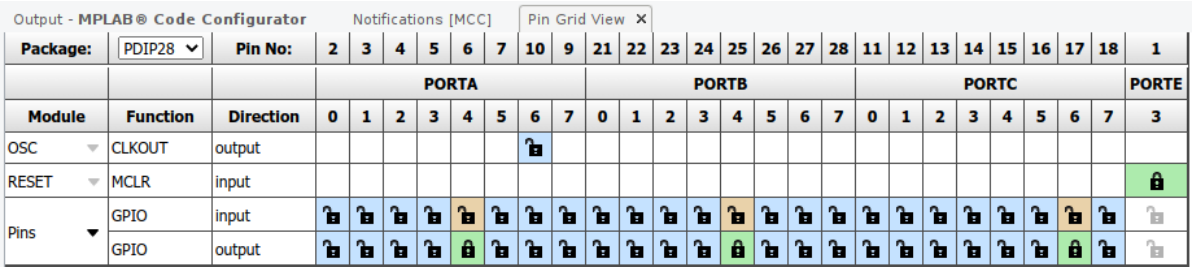


FIGURE 3.3 – Configuration des Leds de la carte PIC.

Modifier les noms des entrées-sorties numériques Dans la zone 3, nous allons renommer nos entrées-sorties numériques pour qu'elles correspondent aux fonctionnalités de l'application. Renommer les LEDs comme celles sur le schéma de la carte, soit :

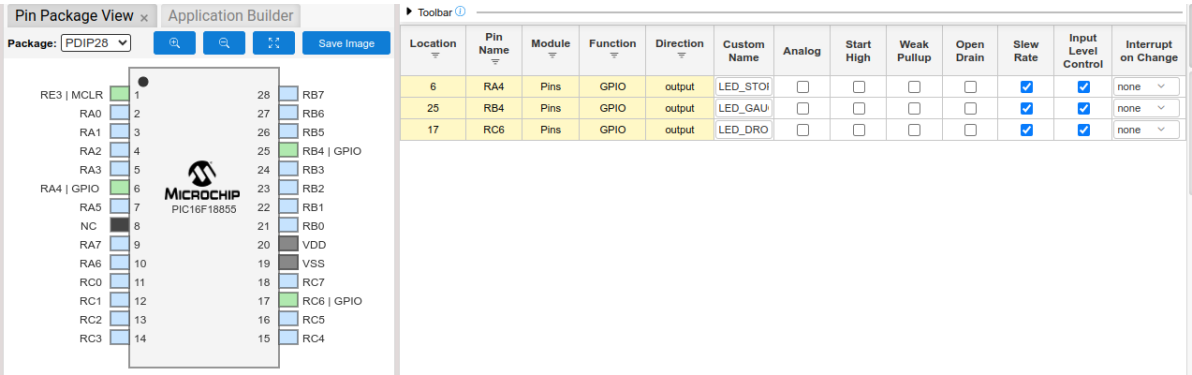


FIGURE 3.4 – Personnalisation des noms des LEDs.

Configurer une entrée numérique Il reste à configurer le bouton poussoir `BP` de la carte. **Compléter** la première ligne du tableau 3.1 à partir du schéma de la carte d'extension. Aller ensuite dans `Pins Grid View` (zone 5) et **repérer** la colonne concernée. Par contre, faire bien attention que la ligne corresponde à `GPIO input` car `BP` est une entrée pour le microcontrôleur ! **Cliquer** à l'intersection pour que le cadenas se ferme.
Pour terminer, personnaliser la broche utilisée par le bouton poussoir. Sur la carte du PIC, vous avez dû remarquer qu'il n'y avait pas de résistance de pull-up. Il faut donc utilisée celle disponible en interne dans le PIC. Pour cela, **cliquer** sur la colonne `Weak Pullup` et **renommer** la broche comme ci-dessous :

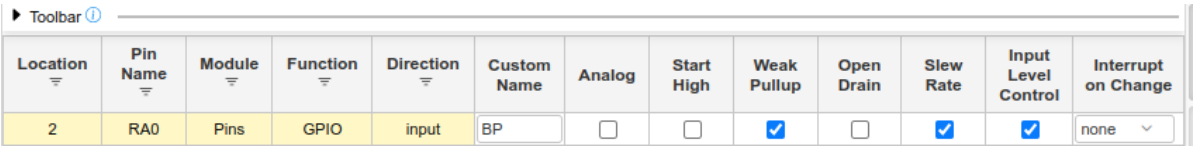


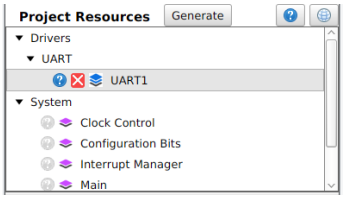
FIGURE 3.5 – Personnalisation et configuration du bouton poussoir.

3.2.2 Configurer la liaison série RS232 du PIC

Pour chaque projet que vous allez réaliser, nous vous demandons de configurer la liaison série. Vous avez vu durant le tutoriel qu'avec l'aide de la commande `printf()` , on dispose d'un moyen d'interagir avec le microcontrôleur. On utilisera cette liaison pour mettre au point tous nos programmes en affichant dans la console des messages.

Comme le circuit n'a pas encore été ajouté à la zone 1, il faut se rendre dans la zone 2 et **cliquer** sur `Drivers` puis `sélectionner` `UART` dans la liste des périphériques de notre PIC. C'est en effet ce circuit qui est en charge de la liaison série RS232 dans un microcontrôleur. **Cliquer** sur le symbole `+` , comme ci-contre.

Il se retrouve désormais ajouté dans la zone 1. Si à un moment donné, vous n'avez plus besoin de ce composant dans votre projet, vous pouvez le supprimer en cliquant sur la croix rouge, il redevient alors disponible dans la zone 2.



Deux nouveaux blocs `UART1` et `EUSART Peripheral` ont été ajoutés dans la zone 3 :

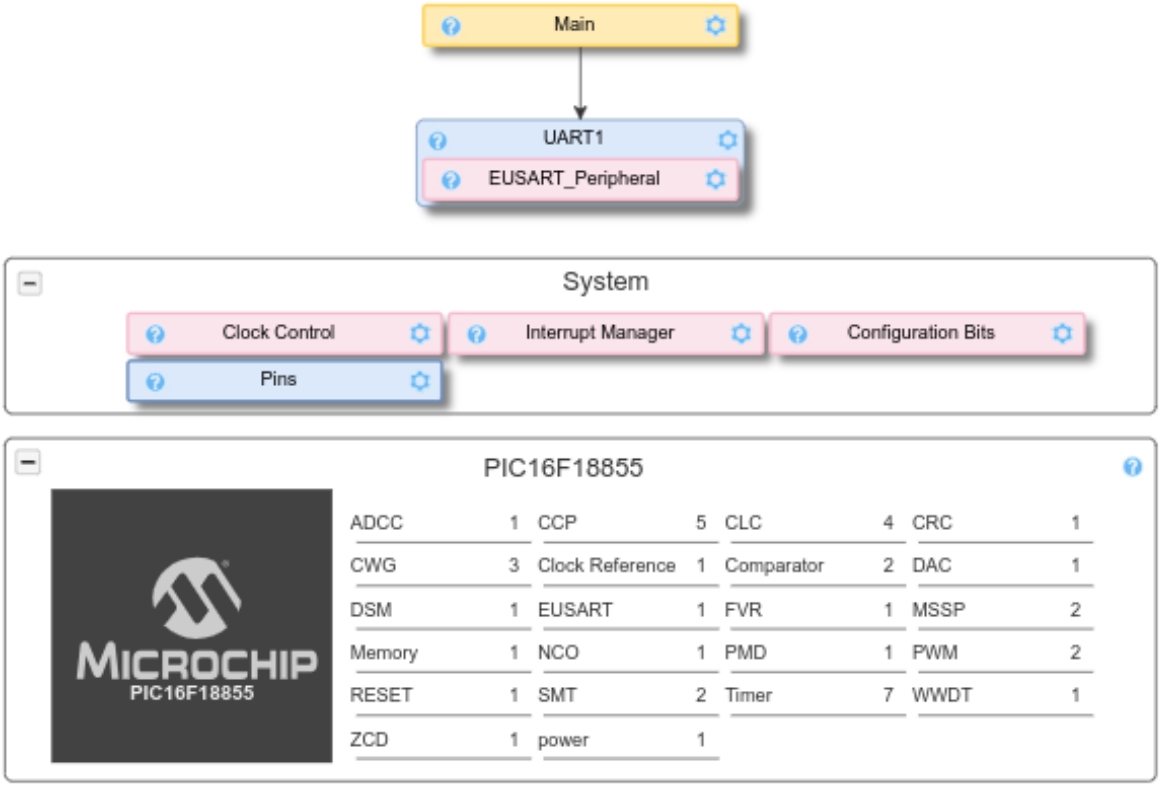
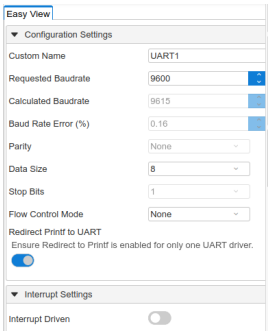


FIGURE 3.6 – Schéma bloc après ajout de l'UART.

La fenêtre de configuration du bloc `UART1` s'ouvre automatiquement à droite. On conserve la configuration par défaut à savoir (vitesse de communication = 9600 bauds, 8 bits de données, pas de parité et un bit de stop). **Activer** `Redirect STDIO to EUSART` comme ci-contre. Cette option nous permet d'utiliser l'instruction `printf()` pour transmettre des messages à la console du terminal.



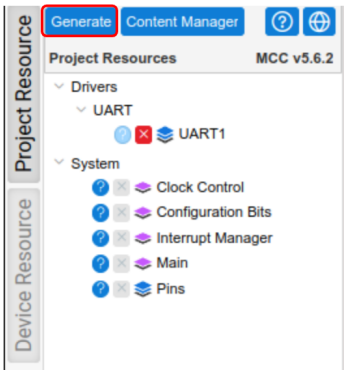
Il nous reste à définir les broches du PIC à utiliser pour l'émission (TX) et la réception (RX) de caractères sur la liaison série RS232. Il faut retourner dans `Pins Grid View` (zone 5). La broche `RC4` est la broche pour l'émission (TX) et `RC5` celle de réception (RX). Vous devez donc obtenir la configuration suivante :

			PORTA								PORTB								PORTC								PORTE
Module	**Function**	**Direction**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**3**
EUSART	RX	input																									
	TX	output																									

FIGURE 3.7 – Configuration des broches d'émission (TX) et de réception (RX) de l'UART.

3.2.3 Générer les fichiers de configuration par MCC

Nous avons enfin terminé de configurer le microcontrôleur PIC avec les périphériques dont nous avons besoin dans notre application. Il reste une dernière étape à effectuer : la génération des fichiers de configuration par le plugin MCC. Cette étape est à réaliser à chaque fois qu'une modification est apportée à la configuration du microcontrôleur, aussi minime soit-elle. Pour générer les fichiers, il vous suffit de cliquer sur le bouton `Generate` de la zone 1.



Dès lors, la zone de notification (zone 5) change d'onglet et bascule sur l'onglet `Output – MPLAB Code Configurator`. Et au bout de quelques secondes, elle vous informe si la génération s'est bien déroulée et quels sont les fichiers qui ont été générés.

```
User Output   Notifications   Pin Grid View ×

Output - MPLAB® Code Configurator ×

21:46:40.689    INFO:  Generation Results
21:46:40.691    INFO: ****************************************************
21:46:40.699    INFO: main.c                                      Success. New file.
21:46:40.701    INFO: mcc_generated_files/system/clock.h          Success. New file.
21:46:40.702    INFO: mcc_generated_files/system/config_bits.h    Success. New file.
21:46:40.703    INFO: mcc_generated_files/system/interrupt.h      Success. New file.
21:46:40.704    INFO: mcc_generated_files/system/pins.h           Success. New file.
21:46:40.706    INFO: mcc_generated_files/system/src/clock.c      Success. New file.
21:46:40.707    INFO: mcc_generated_files/system/src/config_bits.c Success. New file.
21:46:40.709    INFO: mcc_generated_files/system/src/interrupt.c  Success. New file.
21:46:40.710    INFO: mcc_generated_files/system/src/pins.c       Success. New file.
21:46:40.711    INFO: mcc_generated_files/system/src/system.c     Success. New file.
21:46:40.712    INFO: mcc_generated_files/system/system.h         Success. New file.
21:46:40.713    INFO: mcc_generated_files/uart/eusart.h           Success. New file.
21:46:40.714    INFO: mcc_generated_files/uart/src/eusart.c       Success. New file.
21:46:40.715    INFO: mcc_generated_files/uart/uart_drv_interface.h Success. New file.
21:46:40.716    INFO: mcc_generated_files/uart/uart_types.h       Success. New file.
21:46:40.857    INFO: ****************************************************
21:46:40.858    INFO:  Generation complete (total time: 1463 milliseconds)
21:46:40.859    INFO: ****************************************************
21:46:40.859    INFO: Generation complete.
```

FIGURE 3.8 – Informations sur la génération des fichiers de configuration par MCC.

Si l'on veut y voir un peu plus clair dans l'interface, on peut sans aucun souci fermer le MPLAB CODE CONFIGURATOR pour l'instant. Il vous suffit de cliquer à nouveau sur son icône .

La première chose que l'on peut constater dans notre arborescence de projet sous MPLABX est qu'un certain nombre de fichiers ont été créés :

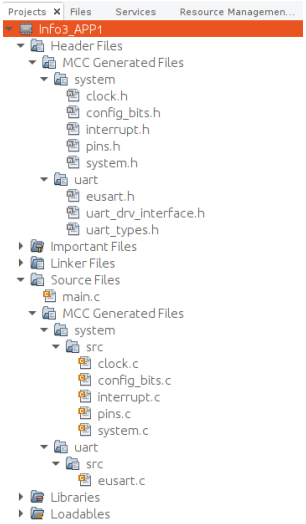


FIGURE 3.9 – Arborescence du Projet après génération des fichiers de configuration par MCC.

Cela peut sembler un peu exagéré étant donné que l'on veut juste faire clignoter une LED, mais sur un projet plus complexe, c'est vraiment confortable d'avoir un fichier C et un fichier d'entête par module activé dans le microcontrôleur.

Mais, même si le projet comporte bon nombre de fichiers, il n'y en a qu'un seul à modifier, c'est le `main.c`.

Attention

Vous ne devez en aucun cas MODIFIER le contenu des autres fichiers !!!
Néanmoins, on vous invite fortement à consulter leur contenu au moment venu.