Rapport de laboratoire

**Ecole supérieure**

Électronique

Laboratoire MINF

Salle R110

**Bootloader PIC32**

**Réalisé par :**

Basile Gaspar

**A l’attention de :**

M.Bovey

**Date :**

Début : 05.05.2022

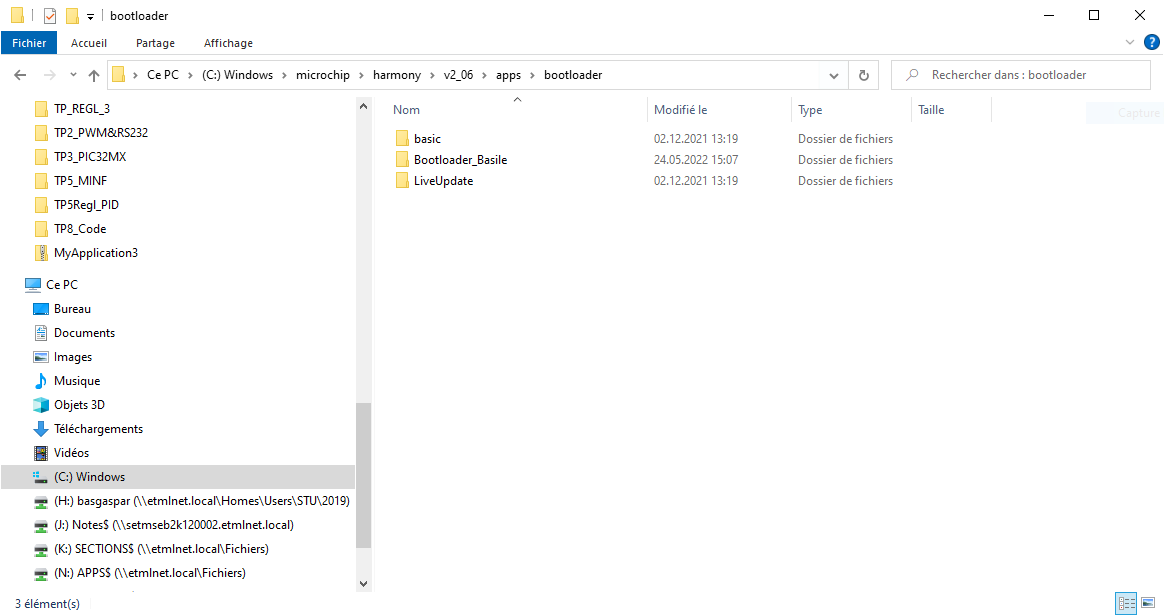
Fin : 06.06.22

# Cahier des charges

Voir annexe 1

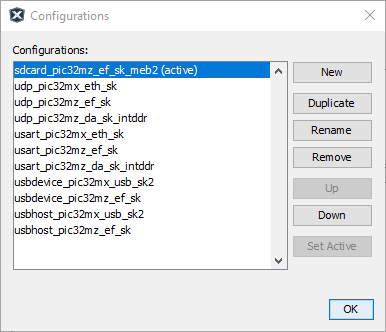
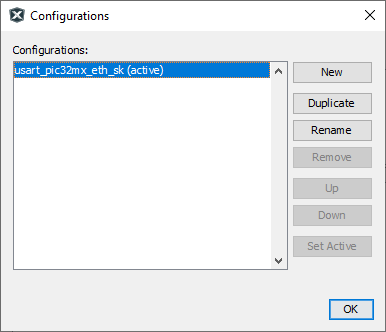
# Création d’un projet et config bootloader

Afin de permettre au kit d’accueilir le bootloader, j’ai commencé par suivre la marche à suivre rédigée par M.Castlodi. (voir références).

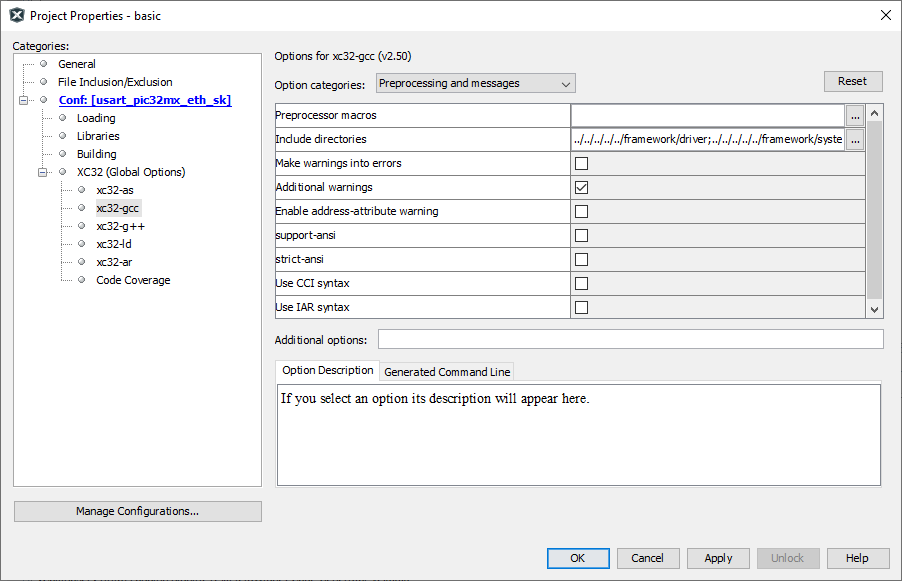


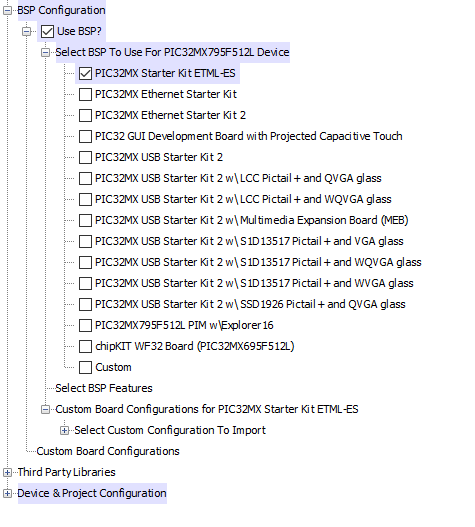
Je commence par me rendre dans C:\microchip\harmony\v2\_06\apps\bootloader puis copie/colle le dossier « basic » en le renommant avec le nom de mon projet. Je laisse ce nouveau projet dans le même répetoire.

J’ouvre ensuite le projet puis active la configuration « usart\_pic32mx\_eth\_sk » et supprime toutes les autres :

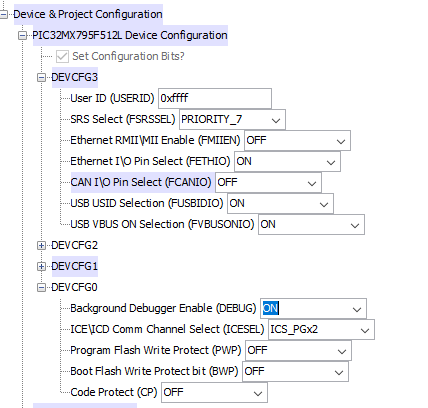
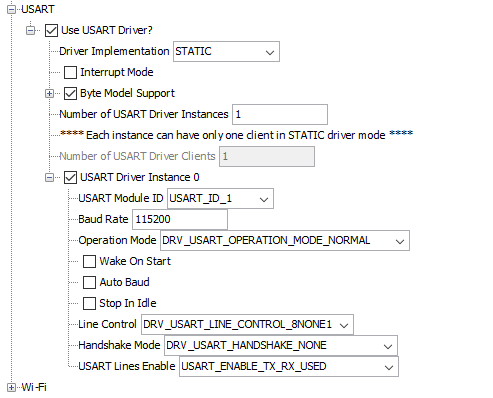


Je règle ensuite l’habituel « additionnal warning ». Attention : Laisser en optimisation 1 pour cette application



Je vais maintenant effectuer les configurations du projet en allant dans Tools > Embedded > MPLAB Harmony Configurator.  
  
Pour commencer je sélectionne le kit micro de l’ETML-ES dans « BSP Configuration » :  


Il s’agit ensuite de modifier le champs « Background debugger Enable » dans DEVCFG0 à « on » et le champs « CAN I/O Pin Select » à « on » dans DEVCFG1.  
Il faut également configurer l’UART1 comme ci-dessous :



On charge ensuite la nouvelle configuration dans l’application :



Nous venons de créer l’application configuré en bootloader prête à charger des fichiers .hex.

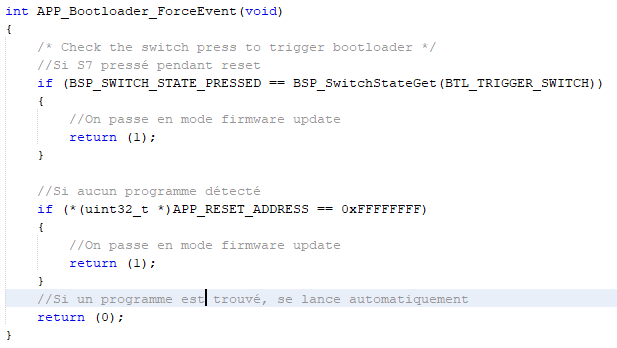
Telle que configurée, l’application est censée faire clignoter la LED D10. J’ai tout d’abord remarqué que ça ne fonctionnait pas, il a en fait fallu régler le port de la LED en question en « GPIO\_OUT » car le BSP habituelle de l’ES n’est pas importé.   
  
En regardant le schéma de notre kit, je vois que la diode D10 est sur la pin 38 du microcontrôleur et correspond à la LED1 :





En chargeant à nouveau le programme, la LED clignote correctement.

Voyant maintenant rapidement comment l’application fonctionne :



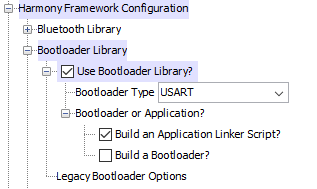
On sépare l’application en deux cas : Soit aucun programme n’est détecté et donc on est sur le mode bootloader avec la LED qui clignote (mode firmware update) soit une application est détectée et on la charge automatiquement au reset.  
  
On peut repasser sur le bootloader en appuyant sur le bouton S7 pendant un reset.

# Configuration de l’application à charger

Il s’agit maintenant de configurer l’application que l’on va charger dans le bootloader.

Je me rends dans la config d’un projet exercice que j’avais réalisé consistant à lire le registre des secondes d’un RTCC pour afficher un compteur s’incrémentant toutes les secondes jusqu’à 60.

On se rend donc dans Harmony Framework Configuration > Bootloader Library et cocher la case « Use Bootloader Library » puis « Build an Application Linker Script » :

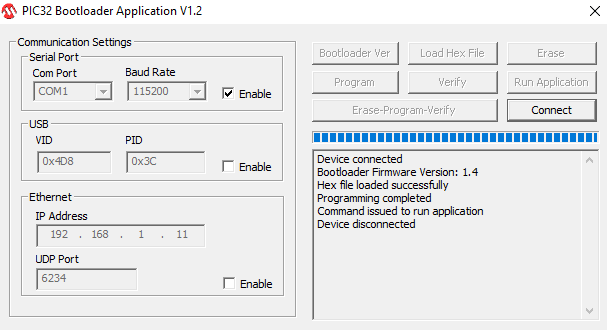


Cette opération va servir à ne pas écraser le bootloader dans l’application que l’on va télécharger.

Une fois ceci fait, on va essayer de charger une application dans le bootloader.

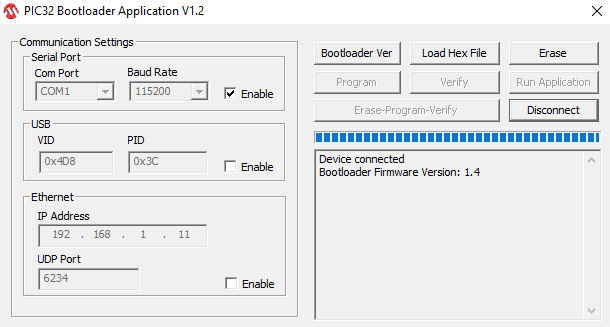
# Chargement d’un programme via application PIC32

Microchip met à disposition une application pour charger un programme dans le bootloader, elle se présente comme ceci :



Trois moyens de communications sont mis à disposition, précédemment, j’avais configuré un UART sur le bootloader donc je vais utiliser la communication série en réglant le port COM et le Baudrate en conséquence.

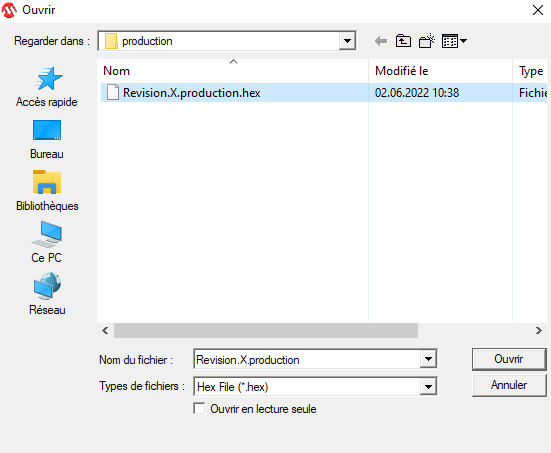
Ensuite, j’appuie sur le bouton « Connect » :

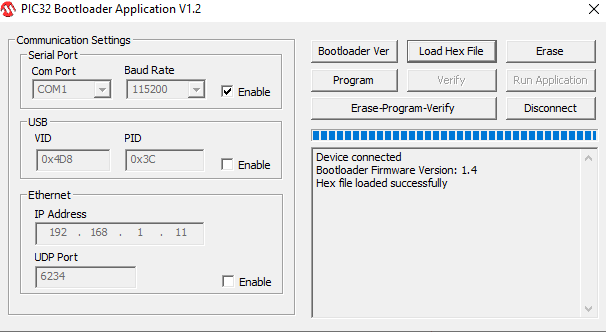


On peut voir que l’application communique bien avec mon kit, la communication série est donc valide. L’application nous fournit également automatiquement la version du Bootloader.  
  
Il s’agit maintenant de charger l’application RTCC précédemment configuré, pour ça je me rends dans le dossier du projet sous le chemin suivant :

*C:\microchip\harmony\v2\_06\apps\MINF\Exercices\Revision\firmware\Revision.X\dist\default\production*

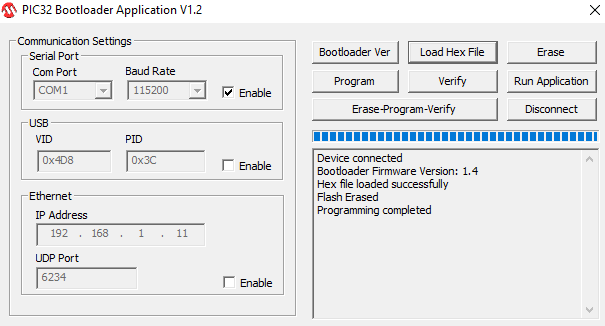
En appuyant sur le bouton « Load Hex File » je séléctionne le fichier .hex du répertoire indiqué :





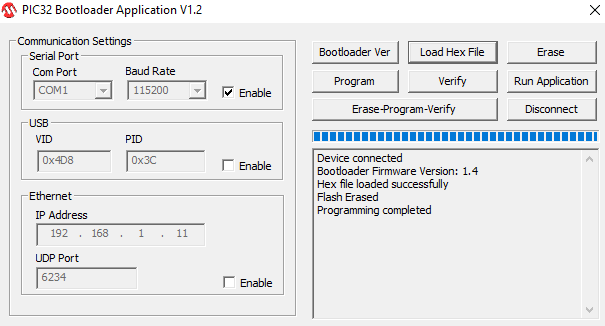
Le fichier s’est chargé avec succès.

Le bouton Erase supprime la mémoire flash précédemment allouée :



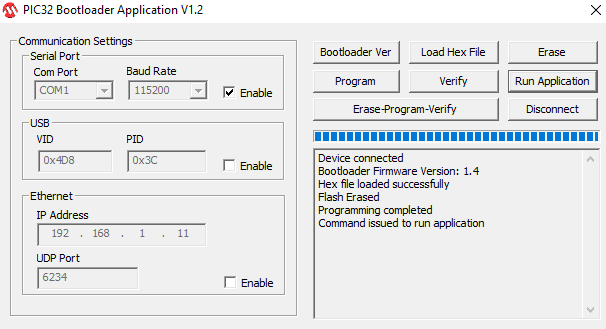
La flash s’est supprimée avec succès.

On programme ensuite le kit avec le bouton Program :



Le kit a été correctement programmé.

Il ne reste plus qu’à appuyer sur Run Application pour lancer l’application chargée sur le kit :



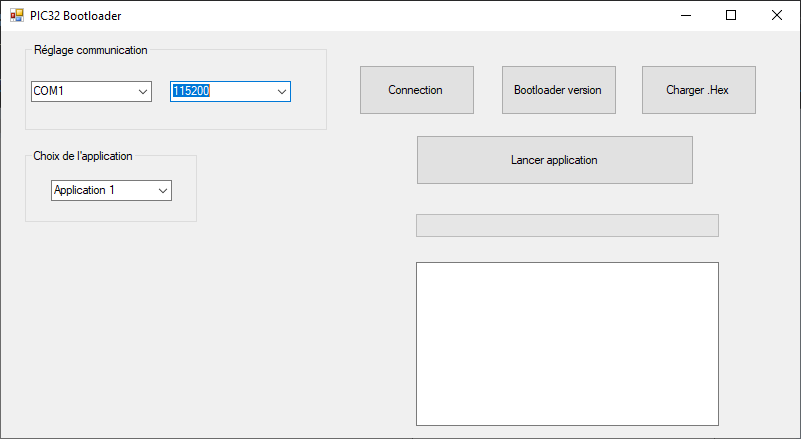
On nous indique que la commande a bien été transmise.

¨

L’application chargée se lance correctement.

# Application C#

Voici la première ébauche de mon application C# ainsi que la description de ses différentes fonctions.



9

7

5

8

6

4

3

2

1

1. L’application communiquera via le port RS232 du kit, à noter qu’il est également possible de faire fonctionner le bootloader en USB, ethernet ou encore via carte SD.  
  
On sélectionnera donc ici le port COM sur lequel se trouvera le kit.

2. On indique ici la valeur du Baudrate. En général, on utilise un baudrate de 115200 pour les communications UART mais il est tout de même important qu’il soit modifiable pour l’utilisateur.

3. On aura le choix entre trois fichiers .hex à charger via bootloader sous les noms provisoires de « Application 1 », « Application 2 » et « Application 3 ».

4. En cliquant sur ce bouton, l’utilisateur tente une connexion avec le kit. Il est obligatoire que la connexion soit validée avant de pouvoir effectuer toutes autres actions sur les autres boutons. Ils seront d’ailleurs grisés avant la connexion établie.

5. On peut obtenir la version du bootloader en cliquant ici.

6. Une fois l’application sélectionnée, on la charge en cliquant ici.

7. Si l’application a été correctement chargée, on peut la lancer sur le kit.

8. Une barre de chargement indique où en est le traitement des informations (chargement de l’app, affichage de la version etc..)

9. Une zone de texte indique à l’utilisateur le bon fonctionnement ou l’échec des opérations.

# Conclusion

Le fonctionnement du bootloader sur le kit de l’ES a pu être mis en place et est fonctionnel. Uniquement la partie graphique de l’application C# ainsi que la sélection des ports COM ont pu être effectuée par manque de temps.

La suite du projet consistera donc à développer l’application afin de pouvoir charger trois applications différentes sur le kit. L’application fournit par microchip pourra être prise comme exemple.

# Références

## Datasheet

Microchip AN1388 « PIC32 Bootloader »

## Autres

Marche à suivre « faire fonctionner le bootloader sur kit » par M.Serge Castlodi

Fichier téléchargé sur : <https://api.256file.com/an1388_source_code_2014_02_14.exe/m-download-410755.html>

Contenant :

-Application de démo bootloader

-Application pour charger .hex « PIC32UBL »

# Annexes

1. Cahier des charges du projet
2. Application C#