**Bootloader**

Rapport technique

A close-up of a microchip

Description automatically generated

Branche : POBJ

Nom du document : 2203\_Bootloader-Rapport\_MRI\_v1.0.0.pdf

Réalisé par : Meven Ricchieri

A l’attention de : Mr. Bovey

Date de début : 9 février 2023

Date de fin : 2 juin 2023

Table des matières

[1 Introduction 2](#_Toc136634424)

[1.1 Avancement 2](#_Toc136634425)

[1.2 Objectif 2](#_Toc136634426)

[2 Documentation 3](#_Toc136634427)

[2.1 Emplacement du bootloader dans le PIC32 3](#_Toc136634428)

[2.2 Communication entre le PC et le MCU 4](#_Toc136634429)

[2.3 Format de fichier HEX 4](#_Toc136634430)

[3 Analyse réelle 6](#_Toc136634431)

[3.1 Méthode 6](#_Toc136634432)

[3.2 Schéma de mesure 6](#_Toc136634433)

[3.3 Mesures 6](#_Toc136634434)

[3.4 Analyse 7](#_Toc136634435)

[4 Algorithme 9](#_Toc136634436)

[5 Conclusion 10](#_Toc136634437)

[6 Sources 11](#_Toc136634438)

[7 Annexes 11](#_Toc136634439)

[7.1 Journal de travail 11](#_Toc136634440)

# Introduction

Dans le cadre de ce projet de POBJ numéro 2203, ma mission consiste à poursuivre le développement d'un logiciel visant à remplacer l'utilitaire existant de Microchip, PIC32 Bootloader Application V1.2. L'objectif principal de ce logiciel est de remédier à une limitation de l'utilitaire actuel, l'absence d'une fonctionnalité permettant de présenter une liste de programmes PIC32 et de permettre aux utilisateurs de choisir celui à introduire dans le microcontrôleur. Le logiciel offrira la possibilité de choisir le mode d'envoi du programme au MCU, que ce soit via USART ou USB.

## Avancement

Ci-dessous se trouve une liste des éléments déjà réalisés :

* Procédure de mise en place du programme de bootloader dans le PIC32, documentée et fonctionnelle
* Procédure de génération de fichiers HEX destinés à un bootloader, documenté et fonctionnelle
* Interface graphique du logiciel PC remplaçant celui de Microchip, réalisée et documentée

## Objectif

L’objectif actuel est donc de réaliser le mécanisme permettant d’envoyer les fichiers HEX au MCU tout en respectant le protocole de communication de Microchip.

J’ai donc commencé par effectuer différentes recherches d’informations sur le bootloader, le type de fichier HEX à envoyer au MCU et le mode d’envoi. En parallèle de cela, j’ai analysé les trames envoyées par le PC lors de l’utilisation du logiciel fourni par Microchip, PIC32 Bootloader Application V1.2, afin de mieux comprendre comment cela fonctionne réellement.

# Documentation

## Emplacement du bootloader dans le PIC32

Les bootloaders de taille réduite sont placés dans la mémoire flash d'amorçage du PIC32. Le fait de placer l'application bootloader dans la mémoire Flash d'amorçage permet à l'utilisateur de disposer d'une mémoire Flash de programme complète pour l'application de l'utilisateur.

Dans le cas de bootloaders dépassant la taille de la Flash de démarrage du PIC32, le bootloader est divisé en deux parties. La table des vecteurs d'interruption (IVT) et le code de démarrage C sont placés dans la Flash d'amorçage, et la partie restante du chargeur d'amorçage est placée dans le programme Flash.

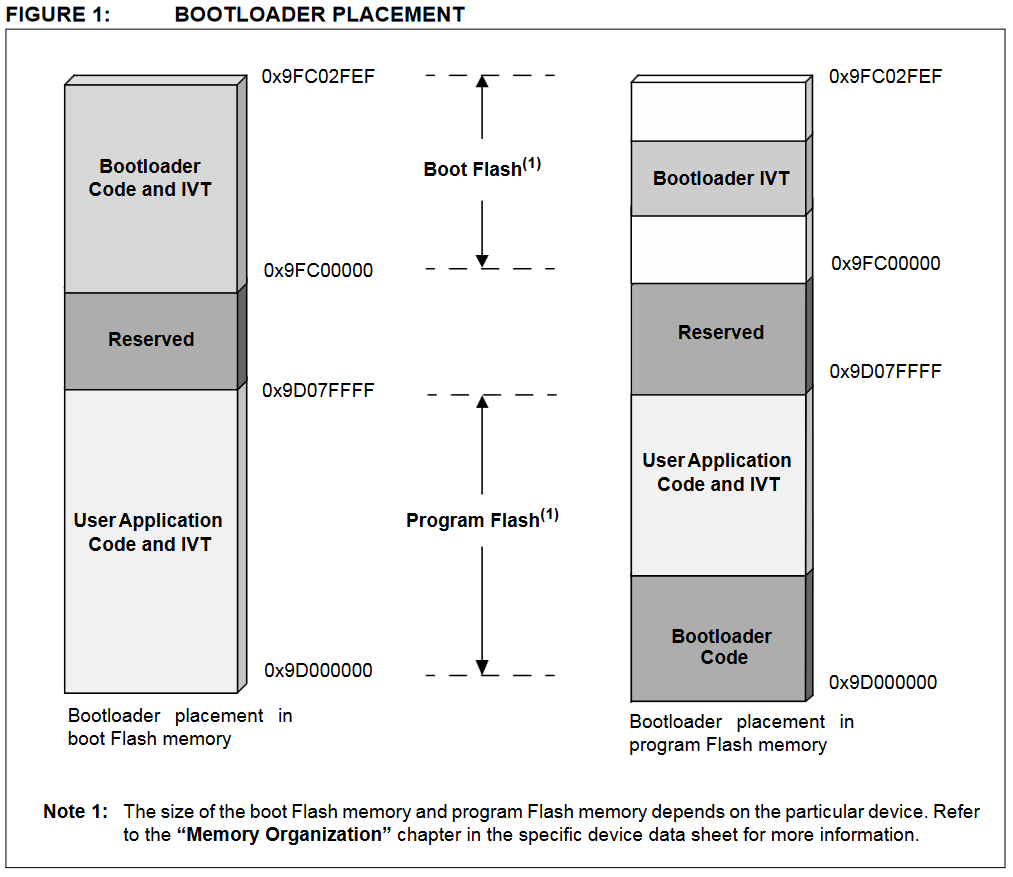


Figure 1 – Bootloader placement

## Communication entre le PC et le MCU

L'application bootloader est implémentée à l'aide d'un framework. Le logiciel du bootloader communique avec l'application hôte du PC à l'aide d'un protocole de communication prédéfini. Le framework de démarrage fournit des fonctions d'interface de programmation d'application (API) pour gérer les trames liées au protocole provenant de l'application PC.

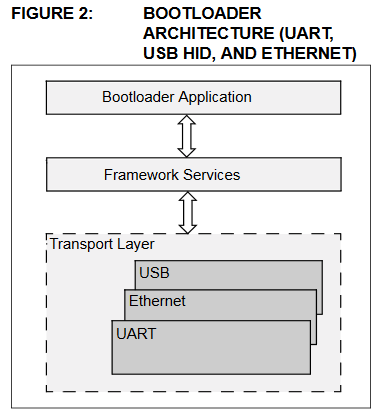


Figure 2

## Format de fichier HEX

Le format de fichier demandé par le bootloader est un type bien connu, il s’agit de l’Intel Hex Format.

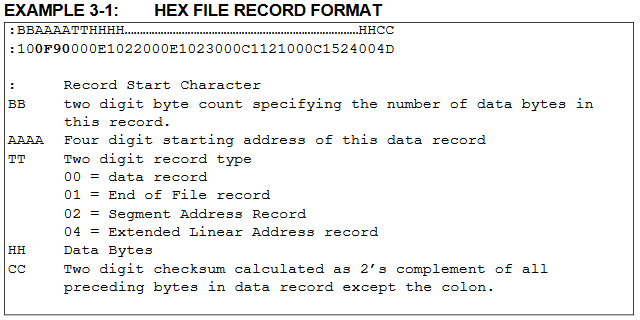


Figure 3

Cependant, le bootloader prévoit des caractères de contrôle supplémentaires qui peuvent poser des problèmes s’ils sont mal interprétés.

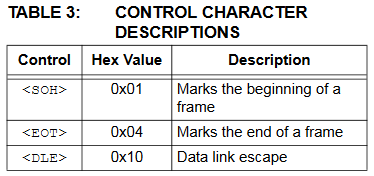


Figure 4

En effet, certains octets du champ de données peuvent imiter les caractères de contrôle SOH et EOT. Le caractère d'échappement de la liaison de données (DLE, Data Link Escape) est utilisé pour échapper à ces octets qui pourraient être interprétés comme des caractères de contrôle. Le bootloader accepte toujours l'octet suivant un <DLE> comme des données et envoie toujours un <DLE> avant tout caractère de contrôle.

L'application hôte du PC peut envoyer les commandes énumérées dans le tableau ci-dessous au bootloader. Le premier octet du champ de données (suivant le caractère de contrôle qui ne fait pas partie des données) contient la commande.

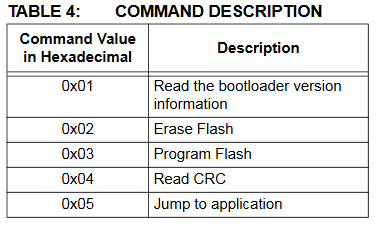


Figure 5

# Analyse réelle

## Méthode

Afin de mieux comprendre la documentation lue, j’ai effectué différentes mesures sur les trames envoyées par l’utilitaire de Microchip au MCU lors de la programmation via USART. J’ai réalisé ces mesures en utilisant un analyseur logique afin de décoder plus facilement les données qu’avec un oscilloscope. J’ai donc branché deux sondes sur l’USART du microcontrôleur, une sur la sortie de TX et l’autre sur l’entrée RX. De ce fait je peux observer tous les signaux transitant sur ces lignes. Une fois l’installation terminé, j’ai lancé la programmation du PIC32 via l’utilitaire de Microchip.

## Schéma de mesure

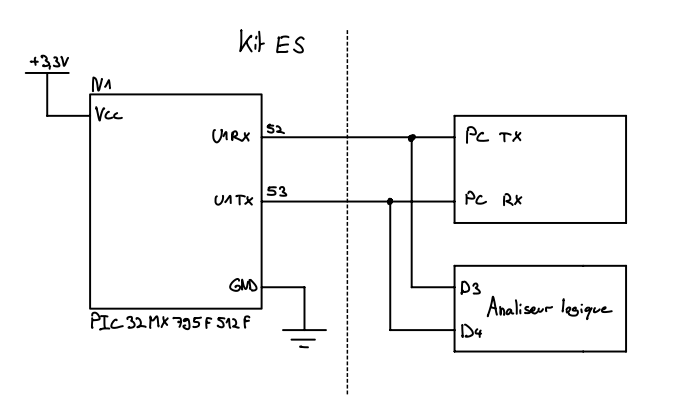
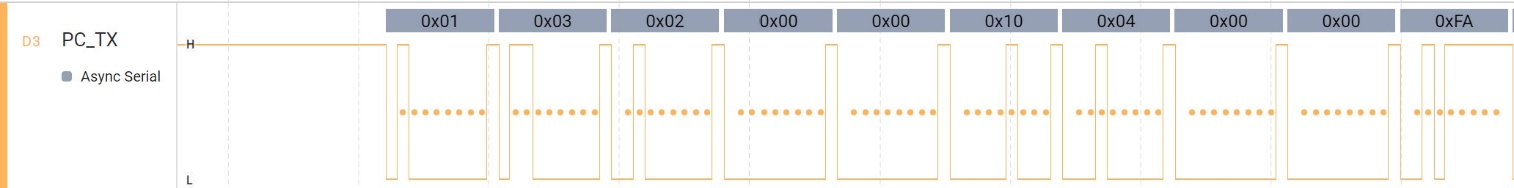


Figure 6 – Schéma de mesure

## Mesures

Data

Data

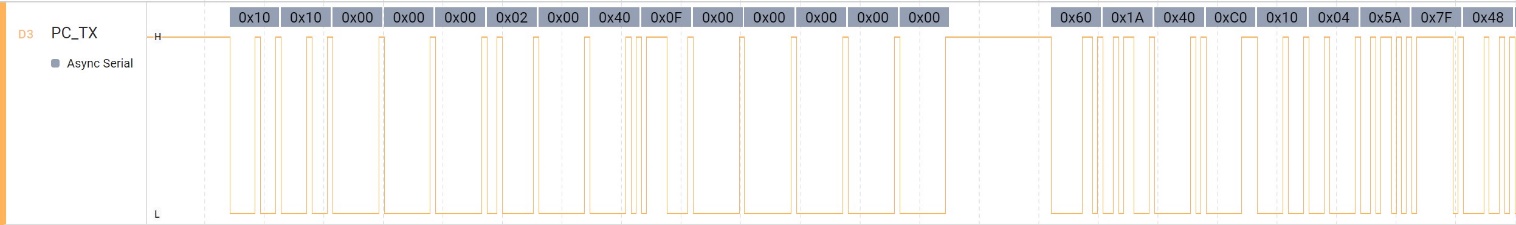


DLE = Data link escape

Program Flash

SOH = Beginning of a frame

Figure 7 - Mesure 1, ligne 1 du fichier HEX



Data

Data

DLE = Data link escape

Figure 8 - Mesure 2, ligne 15 du fichier HEX

## Analyse

Mesure 1



Mesure 3

Mesure 2

A l’aide du fichier HEX du programme, nous pouvons constater que le logiciel effectue quelques traitements avant d’envoyer les bytes au microcontrôleur via l’interface sélectionnée. Dans le fichier HEX, les bytes de contrôle et de commandes ne sont pas présentes. De ce fait le logiciel de Microchip effectue lui-même ces modifications avant de tout envoyer au PIC.

Toutefois, la communication nécessite encore de la recherche, car le PC (host) s’arrête d’envoyer des données tous les temps de temps afin de recevoir une réponse du PIC. Cette partie doit être clarifié avant de pouvoir générer un algorithme permettant de lire le HEX file, le traiter puis tout envoyer au PIC. Je n’ai jusque la pas trouvé la raison pour laquelle il se stoppe.

A picture containing text, screenshot, parallel, rectangle

Description automatically generated

Figure 9 - Trames

A screenshot of a computer

Description automatically generated with low confidence

DLE

??

EOT = Marks the end of a frame

Figure 10 - Mesure 3

Sur cette dernière capture, nous pouvons apercevoir les dernières bytes de data du code HEX suivi de 2 bytes pour l’instant inconnu suivi eux par un caractère de contrôle indiquant au PIC qu’il s’agit une fin de frame. Le PIC répond à chaque fois le même message.

# Algorithme

J’ai tout de même débuté l’élaboration d’un algorithme, permettant de tester le système avant de le mettre en place proprement. Je l’ai créé en Python avec l’aide de mon camarade Ali Zoubir.

Actuellement, ce code permet d’ouvrir un fichier HEX, de trouver les données qui peuvent être comprise comme des commandes et ajouter un DLE en devant.

A screenshot of a computer code

Description automatically generated with low confidence

Figure 11 - Algorithme

# Conclusion

Pour conclure, j’ai effectué diverses recherches afin de trouver des informations pertinentes au niveau du bootloader et du protocole de communication utilisé afin de démarrer correctement le logiciel PC. J’ai également effectué diverses mesures pour confirmer la fiabilité des informations trouvées.

J’ai compris une bonne partie de tout cela mais au niveau du protocole, je n’ai pas entièrement saisi toute la manière de communiquer avec le PIC. Le logiciel de Microchip génère des pauses en envoyant 3 bytes dont 2 sont à déterminer et le dernier est un byte de contrôle de fin de frame.

Malheureusement, en raison de mes nombreux autres travaux, je n'ai pas pu consacrer suffisamment de temps à ce projet sur lequel j'aurais aimé m'investir davantage.

Lausanne, le 2 juin 2023

Meven Ricchieri



# Sources

- <https://ww1.microchip.com/downloads/en/Appnotes/01388B.pdf>

- <https://microchipdeveloper.com/ipe:sqtp-hex-file-format>

- <https://ww1.microchip.com/downloads/en/DeviceDoc/40001779B.pdf>

# Annexes

## Journal de travail

A picture containing text, screenshot, font, line

Description automatically generated