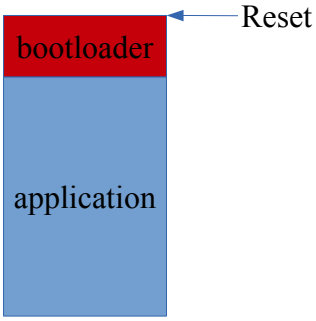
Programmer les PIC18F2550 avec un bootloader

Pour charger un programmer dans un microcontroleur il faut un *outil de programmation*.
Pour les PIC de Microchip, on peut utiliser un **PICKIT** ou un **ICD**. En plus de permettre la programmation du microcontroleur proprement dite, ces outils sont très utiles pour la mise au point d'un programme car il permettent de placer des points d'arrêt et d'examiner la valeur des variables.
Un tel outil est indispensable pour une utilisation professionnelle mais son coût représente un frein pour l'étudiant peu fortuné qui souhaite simplement faire ses premiers pas dans l'apprentissage des microcontroleurs. (Bien que l'on trouve un pickit3 pour moins de 40€).
Ce document explique comment travailler sans programmateur en utilisant un **bootloader**.

Principe

Un ***bootloader*** est un logiciel de petite taille qui est chargé dans la mémoire du pic et qui permet de lire des données arrivant par un canal quelconque (port série, i2c, spi, USB, ethernet,...) et de les écrire dans la mémoire programme du pic. Les données sont en général un nouveau programme...



Au reset, le pic démarre toujours sur le bootloader. Le bootloader teste une condition. Si cette condition n'est pas réalisée le bootloader lance le programme applicatif (fonctionnement normal de l 'application). Si la condition est réalisée alors le bootloader attend des données et écrit les données reçues en mémoire flash. Ensuite on resette le microcontroleur et on repart normalement sur l'application. On vient de mettre à jour le programme.
La condition peut être un niveau 0 sur une broche, l'absence de données pendant un certain temps, la réception d'une séquence particulière, etc...
Le bootloader lui même a été chargé une fois en mémoire avec un programmateur classique. Généralement la zone mémoire dans la laquelle il se trouve a été protégée en écriture car si on écrase le bootloader, on peut plus revenir à un fonctionnement normal sans utiliser à nouveau un programmateur.
Ce procédé est utilisé industriellement pour permettre la mise à jour du firmware par l'utilisateur final. Les firmwares sont disponibles sur les sites des constructeurs, généralement accompagnés d'une application pc spécifique permettant le chargement par un port USB ou Ethernet.

Un bootloader pour le 18F2550

Microchip fournit plusieurs projets de bootloader adaptables pour ces microcontroleurs.
Il faut adapter les sources fournis à sa propre carte (fréquence horloge, type de micro,...) et choisir sa propre condition d'entrée dans le mode programmation.

A l'adresse http://arlotto.univ-tln.fr/pic/pic18/bootloader/ vous trouverez :

- Un projet MPLABX permettant de charger un bootloader USB pour 18F2550 avec quartz de 20MHz.
  La condition d'entrée est RB1=0 au reset.
- Les sources microchip adaptées de ce bootloader (normalement pas nécessaire si vous ne voulez pas changer la condition d'entrée) sous forme d'un projet MPLABX.
- Un projet exemple d'application très simple prévue pour utiliser un bootloader (ou non).
- Les binaires des programmes permettant le chargement de l'application (linux et windows).

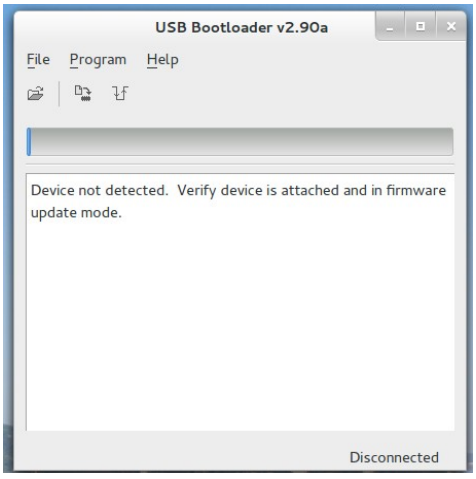
Mode d'emploi

Il faut une carte opérationnelle avec un 18F2550, quartz 20 MHz, liaison USB et RB1 câblé en entrée avec résistance de rappel à l'état haut (de préférence avec un bouton poussoir).
La carte propeller clock répond à ces exigences.

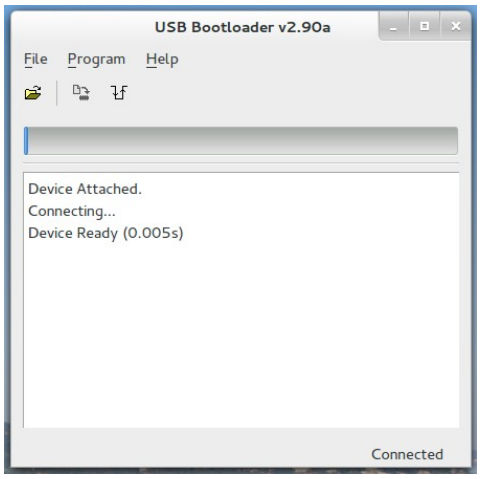
1°/ Vérifiez le bon fonctionnement de votre carte en chargeant un programme classique.
2°/ Ouvrez le projet **18F2550_HID_BootLoader-Bov_RB1_v1.00** et charger le programme avec un ICD3 ou un PICKIT3. Cette opération n'est normalement à faire qu'une seule fois.
Par la suite l'ICD3 ou le PICKIT3 ne seront plus nécessaires.
4°/ Écrivez votre programme applicatif avec MPLAB (on peut à ce stade utiliser MPLAB v8 à condition de générer en mode release) en ajoutant les instructions de décalage de l'adresse de chargement (voir plus loin).
Compiler votre programme et repérer l'emplacement du **fichier hex**.
C'est le fichier à télécharger dans le pic.
Pour MPLABX, il se trouve dans le répertoire :

```
dist/nom_de_la_configuration/production/
```

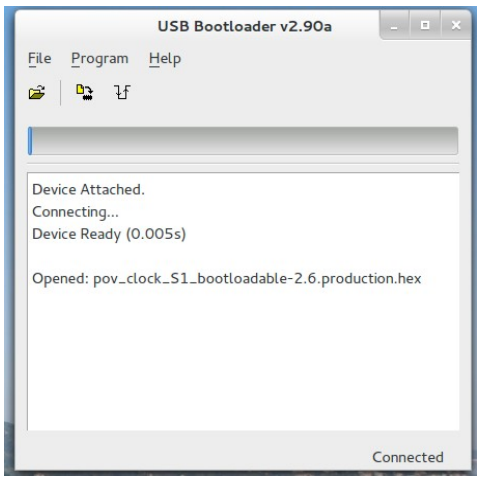
et s'appelle `nom_du_projet.production.hex`
3°/ Lancer le programme **HIDbootloader** correspondant à votre OS.



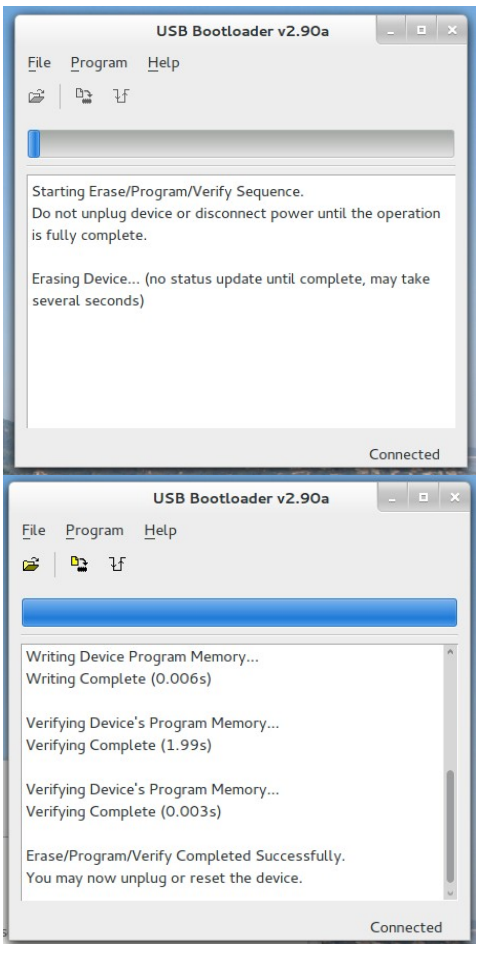
4°/ Connectez la cible en USB en maintenant RB1 à 0 puis relâcher RB1. Le logiciel va reconnaître la cible :



5°/ Charger le fichier hex par le menu File → Import Firmware Image



6°/ Programmer le pic par le menu Program → Erase/Program/Verify device



7°/ Vous pouvez alors débrancher ou resetter votre carte. Au prochain démarrage, l'application sera lancée. Si le fonctionnement n'est pas correct, modifiez votre source dans MPLAB, recompilez et reprendre au point 4°.

Décaler l'adresse de chargement

Il y a quatre conditions à respecter pour écrire un programme qui utilise le bootloader :

1- Le programme applicatif et le bootloader doivent utiliser les mêmes valeurs des bits de configurations. En effet, bien qu'il soit possible de reprogrammer les bits de configurations avec le bootloader, il n'est pas conseillé de le faire car le bootloader pourrait alors ne plus fonctionner.

La configuration est donc celle ci :

```
#if defined(__18F2550)
        #pragma config PLLDIV   = 5    // (20 MHz crystal on POV USB board)
        #pragma config CPUDIV   = OSC1_PLL2
        #pragma config USBDIV   = 2          // Clock source from 96MHz PLL/2
        //#pragma config FOSC     = HSPLL_HS // CPU osc = 48MHz (?)
        #pragma config FOSC     = HS // CPU osc = 20MHz
        #pragma config FCMEN    = OFF
        #pragma config IESO     = OFF
        #pragma config PWRT     = OFF
        #pragma config BOR      = ON
        #pragma config BORV     = 3
        #pragma config VREGEN   = ON            //USB Voltage Regulator
        #pragma config WDT      = OFF
        #pragma config WDTPS    = 32768
        #pragma config MCLRE    = ON
        #pragma config LPT1OSC  = OFF
        #pragma config PBADEN   = OFF
        #pragma config LVP = OFF
        #pragma config STVREN   = ON
#endif
```

2- Il faut décaler l'adresse de chargement du début du programme pour éviter qu'il n'écrase le bootloader. Le bootloader occupe les adresses entre 0x0000 et 0x0FFF. L'adresse 0x0000 est l'adresse de reset qui est capturé par le bootloader. L'application démarre alors en 0x1000.

Pour pouvoir facilement passer d'une version utilisant l'ICD à une version pour le bootloader, le plus simple est de définir un constante et de commenter la ligne pour utiliser l'ICD. (Avec MPLABX utilisez plutôt la configuration de projet)

Placez ceci avant le début de votre code (généralement avant void main(void))

```
#define BOOTLOADABLE // à commenter pour utiliser l'ICD
#if defined(__18F2550) && defined(BOOTLOADABLE)
// Pour utiliser le Bootloader HID de Microchip
#pragma code REMAPPED_RESET_VECTOR=0x1000
```

```
extern void _startup (void);

void _reset (void)

{

   _asm goto _startup _endasm

}

#endif

// mieux vaux générer un warning pour ne pas charger avec l'ICD !

#ifdef BOOTLOADABLE

#warning "Attention Version bootloadable utiliser le bootloader HID pour la
charger "

#endif

#pragma code

void main(void) {

 …

}
```

3- Les vecteurs d'interruptions qui sont normalement en 0x0008 et 0x0018 doivent être décalés en 0x1008 et 0x1018. Toujours avec la constante BOOTLOADABLE on procède ainsi :

```
#include <p18cxxx.h>
//prototypes des gestionnaires d'interruption
void InterruptHandlerHigh (void);
void InterruptHandlerLow (void);

//Placement des sauts vers les gestionnaires d'interruptions
// 0x0008 et 0x0018 pour l'ICD
// 0x1008 et 0x1018 pour le bootloader HID
//-----------------------------------------------------------------------
// High priority interrupt vector
#if defined(BOOTLOADABLE)
   #pragma code InterruptVectorHigh = 0x1008
#else
   #pragma code InterruptVectorHigh = 0x08
#endif
void
InterruptVectorHigh (void)
{
  _asm
    goto InterruptHandlerHigh //jump to interrupt routine
  _endasm
}
// Low priority interrupt vector
#if defined(BOOTLOADABLE)
 #pragma code InterruptVectorLow = 0x1018
#else
 #pragma code InterruptVectorLow = 0x18
#endif
void
InterruptVectorLow (void)
{
  _asm
    goto InterruptHandlerLow //jump to interrupt routine
  _endasm
}
// Routines gestionnaire d'interruption
```

```
//-----------------------------------------------------------------------------
// High priority interrupt routine
#pragma code
#pragma interrupt InterruptHandlerHigh
void
InterruptHandlerHigh ()
{

}
//-----------------------------------------------------------------------------
// Low priority interrupt routine
#pragma code
#pragma interrupt InterruptHandlerLow
void
InterruptHandlerLow ()
{
}
```

4- Il faut charger le programme à partir de `0x1000` et éviter la zone `0x0000-0x0FFF` réservée au bootloader. Pour cela on ajoute au projet le fichier linker script :

`rm18f2550 – HID Bootlad.lkr`.

Ce fichier décrit l'affectation des différentes zones mémoires et leur éventuelle protection :

```
CODEPAGE   NAME=bootloader START=0x0      END=0xFFF          PROTECTED
CODEPAGE   NAME=vectors    START=0x1000   END=0x1029         PROTECTED
CODEPAGE   NAME=page       START=0x102A   END=0x7FFF
CODEPAGE   NAME=idlocs     START=0x200000 END=0x200007       PROTECTED
CODEPAGE   NAME=config     START=0x300000 END=0x30000D       PROTECTED
CODEPAGE   NAME=devid      START=0x3FFFFE END=0x3FFFFF       PROTECTED
CODEPAGE   NAME=eedata     START=0xF00000 END=0xF000FF       PROTECTED
```

On voit bien la zone réservée au bootloader et la zone des vecteurs (reset et interruptions) décalée et la zone réservée au programme (page) à partir de `0x102A`.
Si l'on utiliser pas l'ICD la configuration standard convient et aucun fichier linker n'est nécessaire. (utiliser 'exclude from build' dans MPLABX)
La configuration standard (avec ICD) est alors la suivante :

```
CODEPAGE   NAME=vectors    START=0x0      END=0x29           PROTECTED
CODEPAGE   NAME=page       START=0x2A     END=0x7DBF
CODEPAGE   NAME=debug      START=0x7DC0   END=0x7FFF         PROTECTED
CODEPAGE   NAME=idlocs     START=0x200000 END=0x200007       PROTECTED
CODEPAGE   NAME=config     START=0x300000 END=0x30000D       PROTECTED
CODEPAGE   NAME=devid      START=0x3FFFFE END=0x3FFFFF       PROTECTED
CODEPAGE   NAME=eedata     START=0xF00000 END=0xF000FF       PROTECTED
```