Conception et développement d'un périphérique USB pour le simulateur de vol Microsoft Flight Simulator

Ing. D. ROLIN
ECAM – Bruxelles

Normalement, les instruments de bord du simulateur de vol sont contrôlés par l'utilisation de raccourcis clavier. Ce périphérique permet de ne plus devoir utiliser les raccourcis clavier pour contrôler certains instruments de bord. Sa connexion USB rend son utilisation très simple et automatique. Il ne nécessite l'installation d'aucun programme ou drivers pour fonctionner. Il peut être configuré et mis à jour par un PC. Il permet un réalisme nettement plus grand. Cette plateforme peut servir de base pour créer de nombreux périphériques différents.

Mots-clefs : USB, HID, périphérique, firmware, FMS, MCDU, simulateur de vol, Microsoft Flight Simulator

Usually, board instruments from the flight simulator are controlled by the use of key shortcuts. This device allows controlling some board instruments without having to use key shortcuts. Its USB connexion makes the use of it to be very easy and automatic. It does not need the installation of any program or driver to work. It can be configured and updated by a PC. It allows a much bigger realism. This platform may be used as a base to build a lot of other devices.

Keywords : USB, HID, device, firmware, FMS, MCDU, flight simulator, Microsoft Flight Simulator

1. Introduction

Le projet consiste à concevoir et développer un périphérique USB permettant la gestion d'un Flight Management System pour le jeu Microsoft Flight Simulator en partant de zéro et en terminant avec un produit fini et commercialisable.

Le FMS ou "Flight Management System" est un ordinateur de bord servant à assister un pilote lors de la préparation de son vol et pendant son vol. Il est relié à l'ensemble des instruments de bord et peut les contrôler.

Figure 1 : Le FMS

Le périphérique aura pour but de simuler cet instrument de bord pour les utilisateurs d'un simulateur de vol. Il permettra de ne plus avoir à utiliser la souris ou les raccourcis clavier pour le contrôler.

De tels périphériques existent déjà sur le marché. Ils coûtent cependant plusieurs milliers d'euros. Ce périphérique devra être financièrement abordable pour trouver sa place sur le marché.

Le périphérique ne contient pas d'intelligence orientée "FMS" intégrée. Il n'est pas un simulateur de FMS à lui seul. Il a pour but de rendre l'utilisation d'un logiciel de simulation de FMS plus conviviale et plus réaliste. Il faut donc un logiciel FMS installé sur le PC. Le choix du logiciel de simulation de FMS est laissé à l'utilisateur.

Le périphérique est configuré comme un clavier envoyant des raccourcis clavier vers un logiciel de simulation de FMS tournant sur un PC quand un bouton du périphérique est appuyé. Chaque appui sur une touche du périphérique doit envoyer le raccourci clavier correspondant à cette même touche dans l'application. L'appui sur la touche "a" peut, par exemple, envoyer le raccourci "a" mais il peut aussi être configuré pour envoyer "shift + control + F10". S'il est configuré correctement, il simulera l'appui sur la touche de l'interface dans l'application de simulation de FMS.

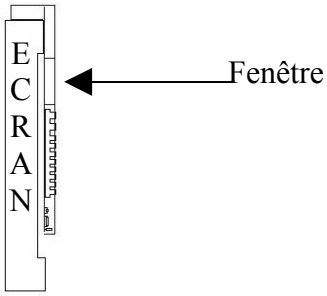


Figure 2 : Support sur l'écran de PC

Il sera posé sur un écran de PC et contiendra une fenêtre transparente permettant de visualiser l'écran du logiciel de simulation de FMS sur l'écran d'ordinateur. L'intelligence et l'affichage sont ainsi déportés sur le PC, permettant de limiter le prix du périphérique.

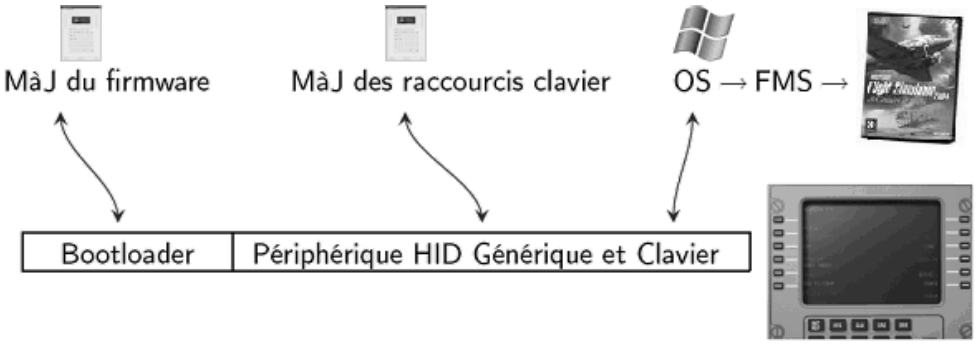
Le simulateur de FMS est un add-on de Microsoft Flight Simulator. Plusieurs firmes proposent le leur. Pour être compatible avec chacun d'eux, les raccourcis clavier envoyés par le périphérique devront être configurables à partir d'une interface graphique sur le PC.

Le design doit être le plus ressemblant possible au véritable FMS du Boeing 737.

Le périphérique doit être Plug & Play. Aucun driver ni programme ne doit être nécessaire à son bon fonctionnement.

La mise à jour du firmware pourra se faire en uploadant une nouvelle version par la connexion USB. Elle nécessitera l'installation du logiciel de mise à jour mais aucun driver ne sera nécessaire. Ce dernier permet ensuite de charger, dans le périphérique, le fichier précédemment téléchargé d'Internet.

Le schéma d'architecture suivant montre les liens entre les différents éléments expliqués ci-dessus.



2. Développement

2.1 Les Bases du Protocole USB

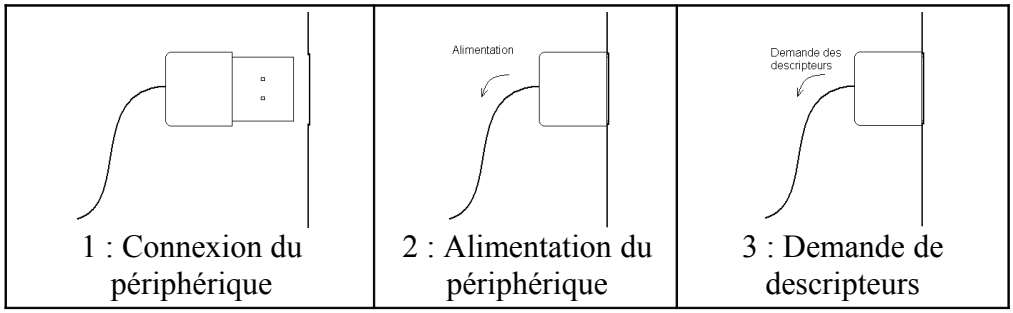
Le fonctionnement du protocole USB est très complexe. Les bases de ce protocole sont nécessaires à la compréhension de ce projet. Celles-ci seront expliquées très brièvement dans ce chapitre. Seuls les éléments nécessaires à la compréhension de ce projet seront abordés. Les couches les plus basses du protocole sont gérées par hardware.

Présentation générale

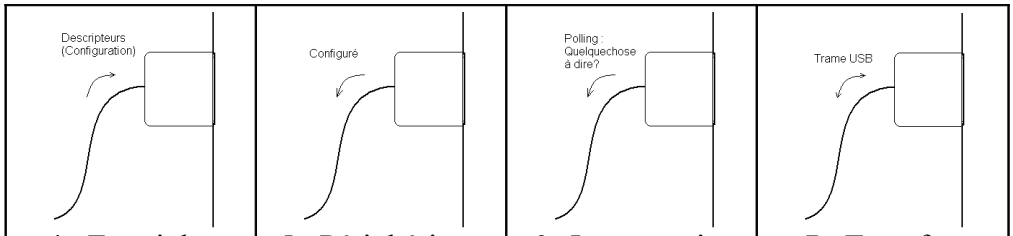
Le bus USB est un bus série utilisant une structure en étoile.
Les ports USB supportent le "Plug & Play". Ceci signifie qu'il suffit de brancher le périphérique. Le système d'exploitation le détectera grâce au changement de tension entre les fils D+ et D-. Il initialise alors le périphérique, puis l'alimente par les fils VBUS et GND. Il envoie une adresse unique au périphérique par l'adresse 0 que le périphérique prend par défaut au démarrage. Le périphérique s'identifie ensuite. Il envoie pour cela des descripteurs contenant la configuration du périphérique. C'est le processus d'énumération. L'OS peut à ce moment charger les drivers du périphérique, à condition que ceux-ci aient été installés. Celui-ci est alors enfin prêt à l'utilisation.

Les ports USB sont capables d'alimenter les périphériques jusqu'à 15 W au maximum par périphérique.

Les grandes étapes



1 : Connexion du périphérique	2 : Alimentation du périphérique	3 : Demande de descripteurs



4 : Envoi des descripteurs (configuration)	5 : Périphérique configuré	6 : Interrogation du périphérique (périodique)	7 : Transfert périphérique - OS.

Les canaux USB

Chaque communication est initiée par le Host Controller qui va interroger chaque périphérique régulièrement. Ceux-ci ne peuvent que répondre. Le Host Controller est le gestionnaire du bus, situé entre le bus USB et l'ordinateur. Le temps entre chaque interrogation est configuré à l'énumération du périphérique. Ce procédé est appelé le polling.

Les périphériques communiquent avec les programmes par l'intermédiaire de pipes ou canaux de communication dont le type, le sens et le format des trames sont fixés à l'énumération.

Il existe deux catégories de pipes :

- Stream pipe ou flux de données : Les données qui le traversent n'ont pas de structure définie. Les stream pipes sont unidirectionnels.

- Message pipe ou canaux de messages : Les données qui le traversent ont une structure prédéfinie. Les message pipes sont bidirectionnels.

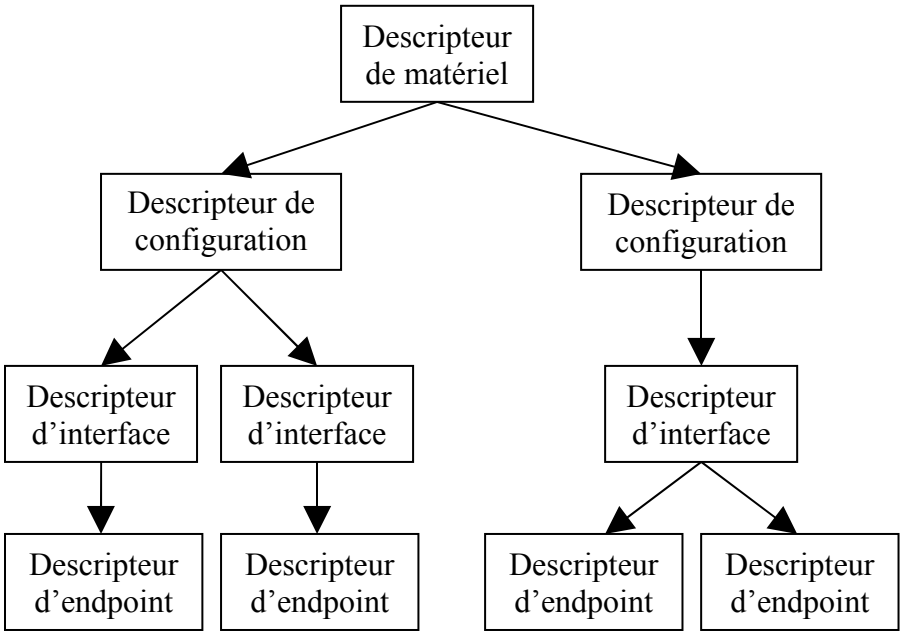
Un périphérique peut contenir plusieurs pipes.
Un périphérique contient au minimum un pipe 0 appelé Default Control Pipe de type message pipe par l'intermédiaire duquel se fait la configuration du périphérique, ainsi que la vérification du statut et le contrôle du périphérique. Les autres pipes sont définis à l'énumération du périphérique et ne peuvent être changés par la suite.

L'énumération

L'énumération a lieu juste après la connexion du périphérique. Celle-ci doit avoir lieu pour que le périphérique devienne utilisable. Le périphérique passe par un certain nombre d'états au fur et à mesure qu'il se configure. Il envoie un certain nombre de descripteurs à l'hôte contenant toutes les informations nécessaires pour être configuré et utilisable.

La structure et les descripteurs

La configuration du périphérique est structurée en plusieurs étages, allant du descripteur de matériel aux descripteurs des trames.
Voici un schéma expliquant cette structure.



Le descripteur de matériel est toujours unique.
Il peut contenir un ou plusieurs descripteurs de configuration. Un périphérique est toujours dans un et un seul de ses états. Il peut passer de l'un à l'autre à condition de se réinitialiser.
Chacun d'entre eux peut posséder un ou plusieurs descripteurs d'interface. Ces différentes interfaces sont chargées simultanément quand le périphérique se trouve dans la configuration. Ceci permet d'avoir simultanément plusieurs périphériques dans un même microcontrôleur.
Chaque interface a au moins un descripteur d'endpoint, qui décrit les message pipes de l'interface. Pour rappel, ceux-ci sont unidirectionnels. Pour obtenir une communication bidirectionnelle, il est indispensable d'avoir plusieurs endpoints sur l'interface.

2.2 Les périphériques HID

Pour ne pas nécessiter d'installation de drivers ou de programmes spécifiques à l'utilisation du périphérique, la classe de périphériques USB HID (Human Interface Device) va être utilisée.
Leurs drivers préinstallés sur tous les OS permettent de connecter un ou plusieurs claviers, souris ou autres périphériques d'interface utilisateur et de les utiliser directement sans devoir installer quoi que ce soit.

Leur utilisation impose de respecter certaines conditions sur les trames envoyées et implique certaines limitations par rapport à la vitesse et au débit de la communication. Ces limitations ne posent aucun problème dans le cas de ce périphérique, ces besoins étant très limités par rapport aux 1.5 Mbits/sec disponible avec cette solution.

2.3 Le design

Le clavier

Le clavier comporte 71 touches en élastomère pouvant être rétroéclairées.

Une matrice de 10 colonnes et 8 lignes a été utilisée pour connecter les touches au microcontrôleur.

L'état des touches est lu par colonne. Les lignes sont mises à l'état haut à tour de rôle. L'ensemble de la colonne (8 bits) est lu en une étape sur le port B. Si un des bits est à l'état haut, c'est que le bouton est enfoncé.

La trame correspondante est envoyée. Celle-ci se trouve dans un tableau. Sa position dans le tableau dépend de la ligne et de la colonne.

2.4 Le schéma électrique

Le microcontrôleur utilisé est le PIC 18f4550. Il dispose d'un périphérique USB intégré, de 32 kb de mémoire flash, de 2 kb de mémoire RAM, de 35 I/O, d'un port ICD (In Circuit Debugging) et d'un oscillateur interne. Cependant, l'utilisation du port USB intégré rend indispensable l'utilisation d'un oscillateur externe supplémentaire.

Autour du microcontrôleur se retrouvent les éléments suivants :

- l'horloge 20 MHz,
- le port USB,
- le port ICD (In Circuit Debugging),
- les capacités de découplage,
- la matrice de clavier (non représentée sur le schéma ci-dessous).

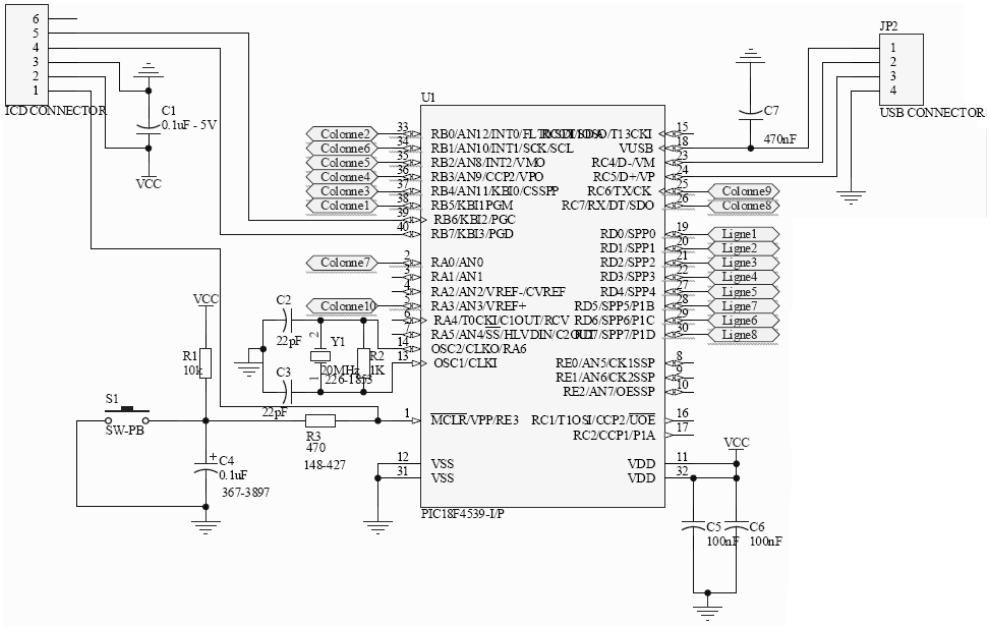


Figure 3 : Schéma électrique partiel

Etant donné la complexité du tracé et des pastilles, il a fallu faire le PCB en double face.

2.5 Le programme embarqué

Ce programme a été écrit en langage C sur Microchip MPLAB IDE V7.21.

Pour charger un premier programme dans le microcontrôleur, le programmeur ICD 2 de Microchip est utilisé.

Figure 4 : ICD 2

Il s'agit d'un débuggeur pouvant être connecté directement sur le prototype.

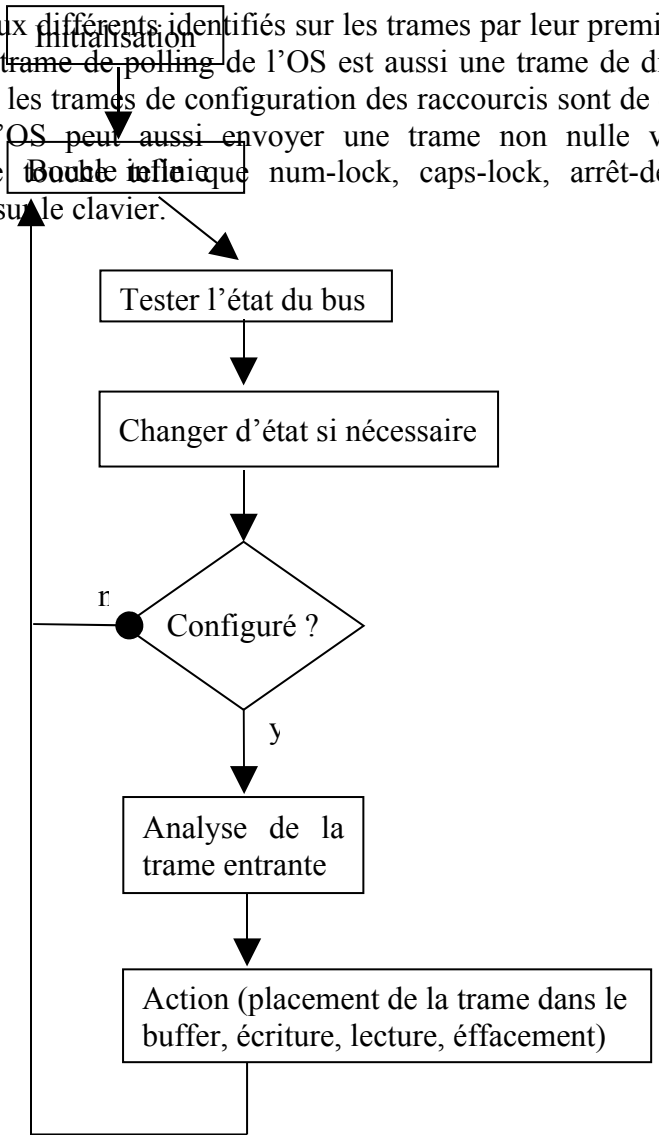
Le programme principal

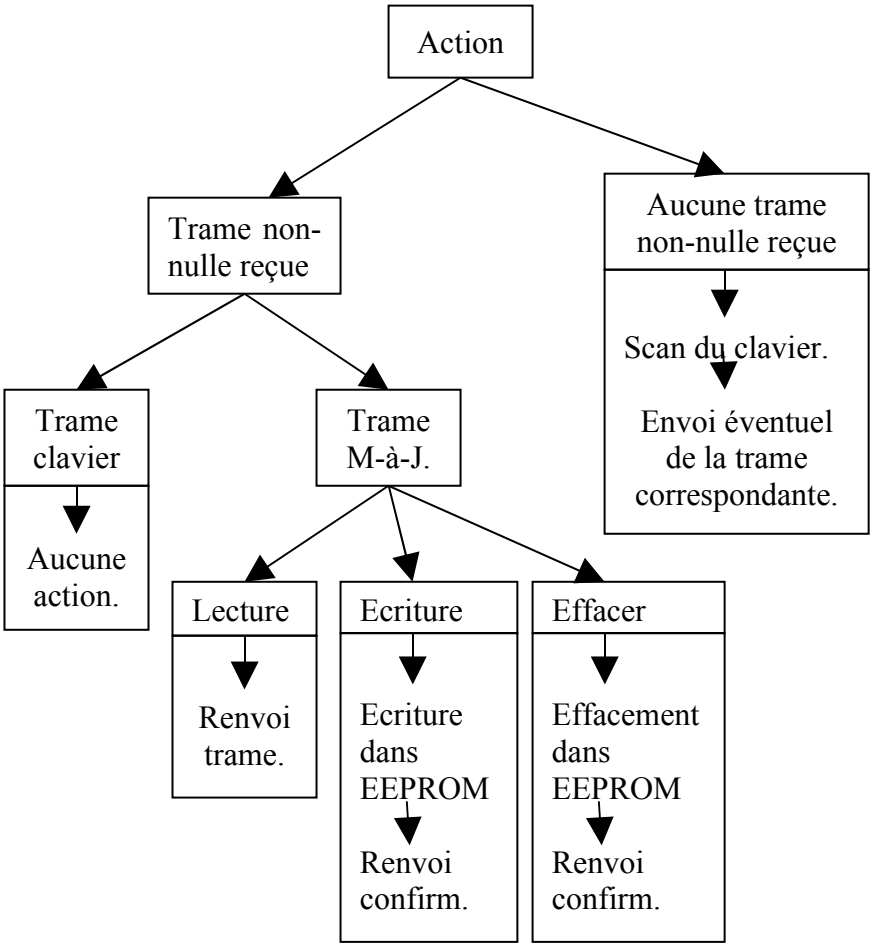
Le programme principal a pour rôle de configurer son périphérique USB dès qu'il est lancé. Il doit ensuite constamment tester l'état du bus pour détecter une erreur ou une déconnection. Après s'être initialisé, il effectue une boucle infinie qui teste l'état du bus, configure le périphérique s'il ne l'est pas. S'il est connecté, il doit analyser la trame entrante de manière à détecter l'action souhaitée.

Lorsqu'une touche est enfoncée, il n'envoie pas immédiatement la trame mais il la place dans le buffer de sortie. Parallèlement, à chaque polling de la part de l'OS, il renvoie le contenu de ce buffer. Si aucune touche n'a été enfoncée, il renvoie une trame remplie de zéros.

Si la trame est une demande de la part du logiciel de configuration, l'action correspondante sera exécutée.

Ces deux types de trames sont différenciés par la valeur du premier octet de la trame. En effet, celles-ci sont envoyées vers deux périphériques différents, actifs simultanément sur le même microcontrôleur. Ils utilisent deux canaux différents identifiés sur les trames par leur premier octet. Dans ce cas, la trame de polling de l'OS est aussi une trame de dimension nulle tandis que les trames de configuration des raccourcis sont de dimension non nulle. L'OS peut aussi envoyer une trame non nulle vers le clavier lorsqu'une touche telle que num-lock, caps-lock, arrêt-défilement sont appuyées sur le clavier.





Le bootloader

Une autre fonctionnalité implémentée dans le périphérique est la possibilité de mettre à jour le firmware par l'USB sans devoir rebrancher l'ICD. Ceci est réalisé par le bootloader.

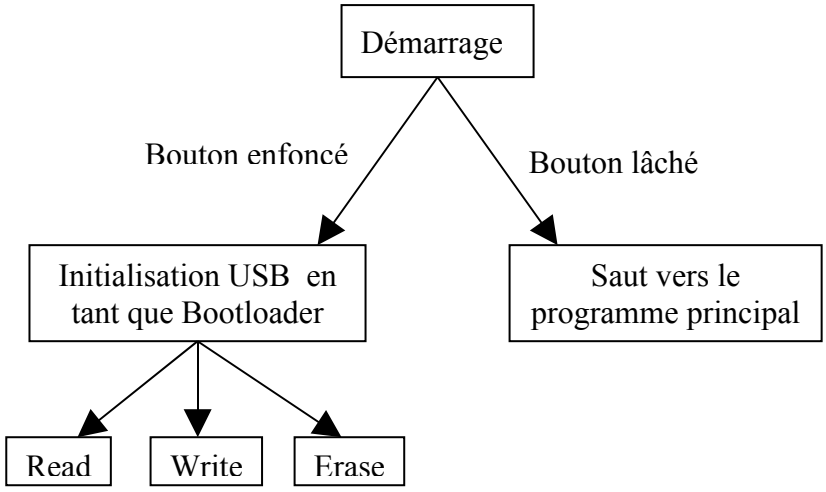
Le bootloader est un programme qui est lancé à chaque démarrage du périphérique. Il est totalement indépendant du programme qui exécutera la tâche principale du périphérique, vu ci-dessus.

Cette très intéressante fonctionnalité est possible grâce à la capacité ISP du microcontrôleur : In System Programming. Ceci signifie qu'un programme dans le microcontrôleur sait écrire dans sa propre mémoire.

Il peut donc modifier son propre code source et se reprogrammer sans aucun matériel spécifique.

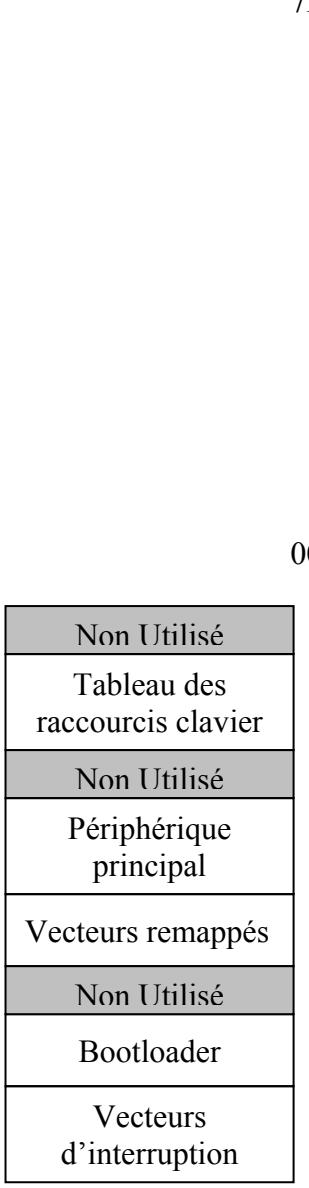
A son lancement, il vérifie une condition (l'appui sur une touche par exemple). Si la condition est remplie (le bouton est enfoncé), il va lancer sa procédure permettant de télécharger un nouveau programme sur le microcontrôleur. Sinon, il lancera le programme qui est chargé juste derrière lui dans la mémoire : le programme principal du périphérique. Si l'utilisateur n'appuie sur aucune touche du périphérique au démarrage, il ne s'apercevra pas de la présence du bootloader.

Ceci permet à l'utilisateur de mettre à jour son périphérique et est très utile lors de la phase de développement du produit.



Etant donné que le bootloader est un programme totalement différent du programme principal, il est nécessaire d'utiliser des vecteurs d'interruption différents pour les deux. C'est pourquoi il convient de remapper les vecteurs d'interruption lors du passage du bootloader au programme principal. Ceci est fait en utilisant une version particulière du fichier linker lors de la compilation du projet et en mettant les bonnes valeurs à certaines variables dans le programme principal et dans le bootloader.

Voici à quoi ressemble l'utilisation de la mémoire avec un bootloader :



Le programme principal est programmé par le port USB à l'aide de ce bootloader. Pour envoyer ce fichier, une fonction sera prévue dans le logiciel de configuration du FMS.

2.6 Le programme PC

Le programme PC sert à configurer le périphérique pour le rendre compatible avec n'importe quel logiciel FMS. Il n'est pas nécessaire au bon fonctionnement du périphérique. Une fois le périphérique configuré, le programme FMS ne doit plus être installé.

Il comporte une interface semblable à un FMS. Quand un des boutons est cliqué, la combinaison de touches qui sera renvoyée par la touche correspondante peut être choisie. Quand le bouton ok est cliqué, le programme envoie la nouvelle configuration au périphérique qui la mémorise dans sa mémoire flash. Ainsi, cette opération ne devra plus être refaite par la suite.

La configuration peut être sauvée dans un fichier "fms" et rechargée par la suite. L'ensemble de la configuration pourra ainsi être refait en 2 clics par la suite. Ceci permet aussi de livrer le produit avec plusieurs fichiers de configuration par défaut.

L'ensemble de la configuration peut aussi être lu à partir du périphérique. Ceci permet de vérifier quels sont les raccourcis configurés.

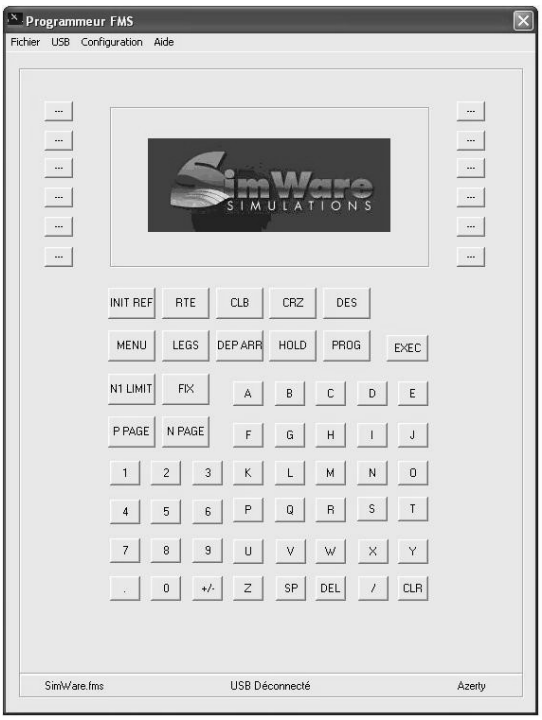


Figure 5 : Fenêtre principale du programmeur FMS

Figure 6 : Fenêtre de configuration d'un raccourci du programmeur FMS

Guide du programmeur

Un premier outil que le programmeur trouvera très utile pour la visualisation de la connexion et des trames USB, est le mode "DEBUG" du programme.
Ce mode, accessible uniquement au programmeur, ouvre un volet à la droite de la fenêtre principale comportant un certain nombre de boutons, cases à cocher et fenêtres de texte utiles lors du développement et du debugage.
Lors d'un accès à une fenêtre de configuration d'un raccourci, le mode débug permet aussi d'avoir quelques contrôles supplémentaires.

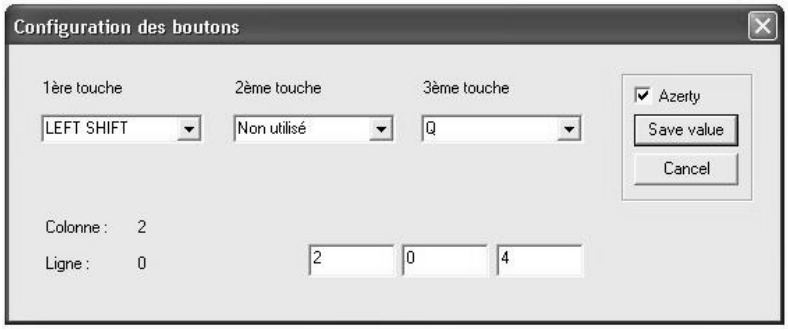


Figure 7 : Fenêtre de configuration d'un raccourci en mode débug

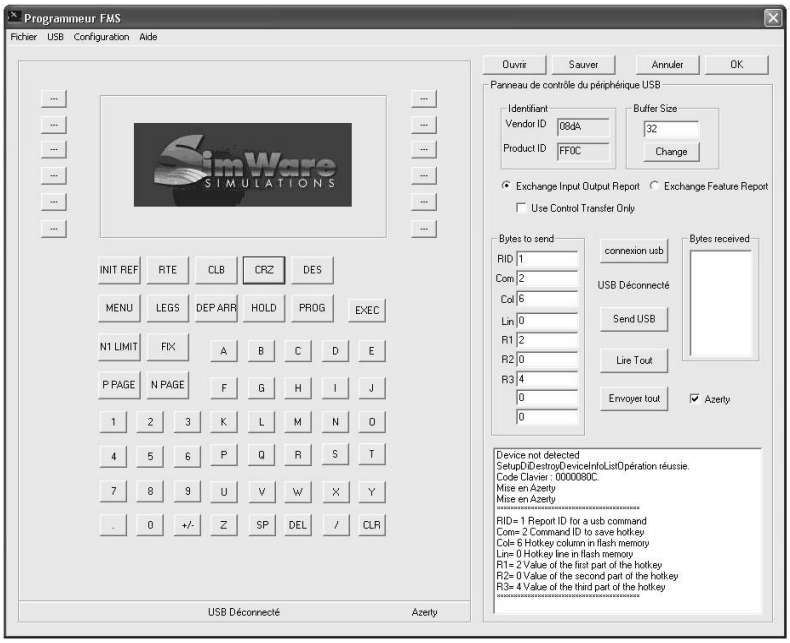


Figure 8 : Fenêtre principale en mode débug

Mise à jour du firmware

La fonction de mise à jour du firmware exécute trois tâches : il analyse le fichier hex, transforme son contenu en trames USB et les envoie sur le bus USB. Le fichier à envoyer est au format hex.

Le code source

Ce programme a été écrit en langage C++ sur Microsoft Visual C++ 6.0.

Pour communiquer par le port USB, il utilise une dll contenant des fonctions de bas niveaux permettant d'ouvrir, fermer, lire et écrire sur le bus USB.

Pour vérifier si le périphérique voulu est présent, cette dll fournit une méthode permettant de disposer de la liste des couples VID/PID des périphériques actuellement connectés. Il suffit de trouver dans cette liste le couple correspondant à ce périphérique et de vérifier qu'il est possible de s'y connecter.

Après avoir chargé cette dll et s'être connecté au périphérique, il faut ouvrir les pipes de communication. C'est à ce moment qu'il est possible de créer et envoyer des trames, ainsi que recevoir et analyser les trames reçues.

Après utilisation, les pipes doivent être fermés.

Le programme tente aussi de détecter automatiquement le type de clavier configuré dans Windows.

2.7 La Communication USB

Les périphériques HID

Les périphériques HID ou Humain Interface Device sont une classe de périphériques USB adaptée aux interfaces entre l'utilisateur et la machine.

L'utilisation de cette classe implique quelques limites de bande passante et de vitesse de transfert mais cela n'est pas un problème dans le cas d'un clavier, surtout que cette limite est de 1,5 Mb/s.
Le besoin en bande passante de ce périphérique est de 13 kb/s.

L'énorme avantage qu'ils apportent est l'utilisation de drivers intégrés et préinstallés sur tous les OS. C'est du véritable Plug & Play. Les périphériques HID sont divisés en plusieurs catégories dont les claviers, les souris et les périphériques génériques.

Deux seront utilisés : le clavier et le générique.

L'utilisation du type clavier permet à Windows de reconnaître automatiquement les trames clavier et de les transférer vers le programme actif. Il en protège cependant l'accès à partir d'un programme. Cela empêche le logiciel de configuration des raccourcis clavier d'envoyer des trames vers ce périphérique pour programmer le raccourci clavier associé à une touche. C'est pour cette raison qu'un périphérique HID générique a dû être utilisé simultanément pour effectuer cette opération. Il ne nécessite pas l'installation d'un driver non plus mais il n'est pas protégé en accès par Windows. C'est donc par l'intermédiaire de ce périphérique que les raccourcis clavier seront configurés.

Le couple VID (Vendor IDentifier) / PID (Product IDentifier) est un couple d'octets unique pour chaque périphérique. Il permet au système d'exploitation de reconnaître un périphérique dès qu'il est branché. Si ses drivers ont déjà été installés, le système d'exploitation les chargera automatiquement. C'est aussi par l'intermédiaire de ce couple qu'un programme peut se connecter et communiquer à un périphérique.

Deux périphériques cohabitent dans un même microcontrôleur.
Leur couple VID / PID est unique pour les 2.

Voici le schéma de la configuration des descripteurs de ce système.

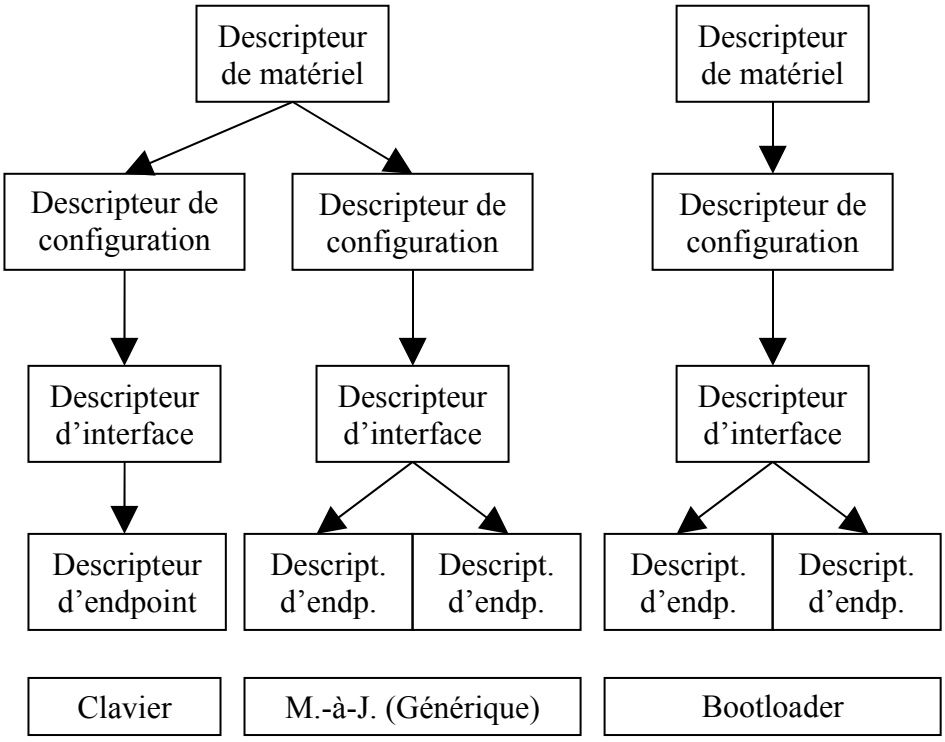


Figure 9 : Configuration du système

A gauche se trouve le clavier. Celui-ci ne possède qu'un seul pipe décrit par l'endpoint descriptor.

Un pipe est un canal de communication unidirectionnel. Son sens est défini par rapport au host controler, c'est-à-dire par rapport au PC. Le clavier envoie des informations vers l'ordinateur. C'est donc un pipe de sens IN. Les trames sont envoyées par le système d'exploitation pour interroger le clavier au travers du Default Control Pipe.

Au centre, se trouve le périphérique HID générique servant à configurer le périphérique clavier. Celui-ci reçoit et envoie des données. Il possède donc deux pipes, un de sens IN et un de sens OUT.

Enfin, à droite, se trouve le périphérique bootloader. Celui-ci est totalement indépendant des deux autres. Il n'est en effet jamais configuré en même temps que les deux autres et se trouve sur un programme différent dans la

mémoire du microcontrôleur. Il possède aussi 2 endpoints, un IN pour lire la mémoire du périphérique, l'autre, OUT pour y écrire.

Les trames ont le format suivant :

Offset	0	1	2	3-23
Field Name	N° de périphérique	Code de résultat / commande	Longueur des données	Données

2.8 L'intégration mécanique

Le système possède un boîtier à l'avant qui viendra sur les côtés. Le pcb sert de fond et doit donc être vissé au boîtier avant. Le pcb doit être suffisamment rigide pour supporter l'appui sur les touches. Si la fixation avec le boîtier avant ne suffit pas à le rendre suffisamment rigide, l'utilisation d'autocollants en caoutchouc venant s'appuyer sur l'écran sans l'abîmer est envisageable.

La face avant est réalisée en plastique. Un rebord sur le haut permet de venir s'appuyer sur le haut de l'écran. Un renfoncement dans les bords permet de placer le périphérique sur les anciens écrans CRT.

Voici un aperçu du fichier AutoCAD.

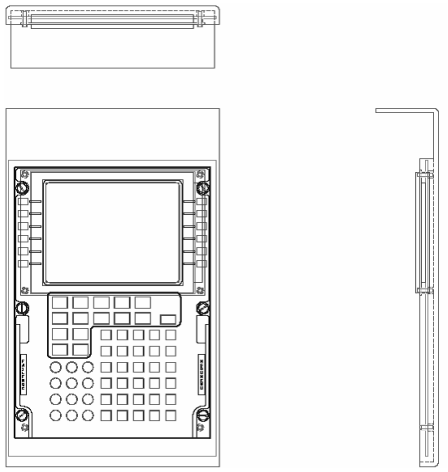


Figure 10 : La face avant sur AutoCAD

Voici quelques images du produit fini en pleine action.

Figure 11 : Le produit fini

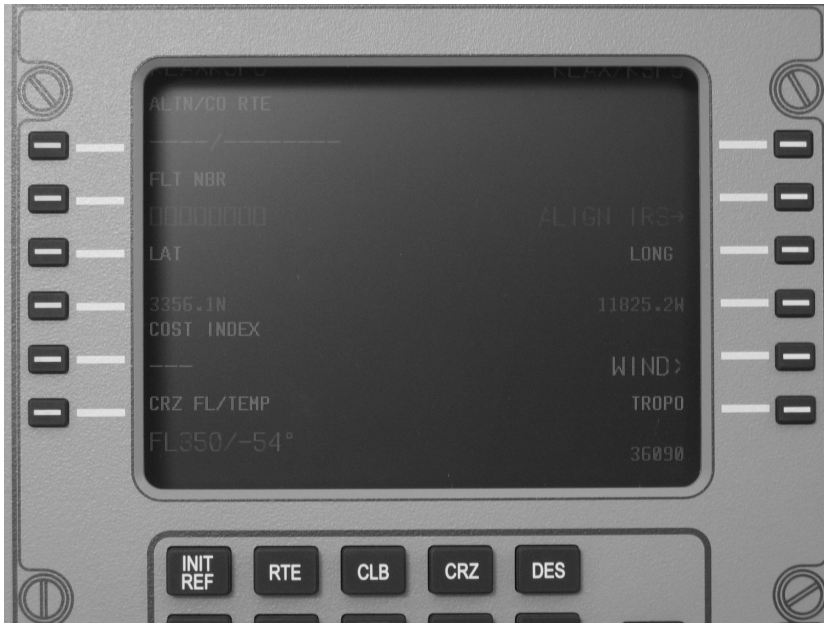


Figure 12 : L'écran du produit fini

L'Alimentation

Le périphérique est entièrement alimenté par le bus USB. Sa consommation ne devrait en effet pas dépasser les 100 mA d'après une rapide estimation.

2.9 Les évolutions possibles

Les Améliorations

Plusieurs améliorations sont possibles dans le futur.

Il est envisageable que le système simule, en plus du clavier, une souris. Le logiciel de configuration permettra de sauvegarder une position sur l'écran. Lorsqu'une touche sera appuyée sur le périphérique, celui-ci simulera un mouvement de souris vers le bouton correspondant du logiciel suivi d'un clic. La principale difficulté sera que le périphérique ne connaît pas la position actuelle de la souris à l'écran. Il devra donc déplacer le curseur de la souris d'une grande distance dans une direction oblique de manière à se retrouver dans le coin de l'écran. A partir de là, il n'a plus qu'à se déplacer de la distance souhaitée dans la direction donnée et effectuer un clic. Ceci permettrait de rendre le périphérique compatible avec les logiciels de simulation de FMS n'utilisant pas de raccourci clavier.

Une autre amélioration possible est le rétroéclairage du clavier. Ceci nécessite d'utiliser plus de 100 mA, ce que tous les ports USB ne peuvent pas fournir. Il est donc indispensable de pouvoir fonctionner sans le rétroéclairage si le courant maximum fourni par le port est trop faible. Une autre solution est de permettre l'utilisation d'une alimentation externe. Ceci a pour principal inconvénient d'augmenter le prix et de diminuer la facilité d'utilisation.

Une évolution possible du logiciel de configuration est l'automatisation de téléchargement du firmware sur Internet. Ceci permettrait de sécuriser l'opération en évitant le téléchargement de fichiers non valides.

Système Polyvalent

Enfin, le nombre de nouveaux périphériques qu'il est possible de construire sur base de cette plateforme USB est infini. Ce genre de périphérique est imaginable pour n'importe quel jeu ou programme.

3. Conclusion

Un des plus grands points forts de ce projet est sans aucun doute la possibilité d'utiliser le produit sans avoir à installer de drivers.
Ceci est possible grâce à l'utilisation du protocole USB HID.

Un autre point fort est l'utilisation du bootloader permettant de changer le programme embarqué sans nécessiter de programmeur. Cela permet de proposer des versions mises à jour du programme aux clients et, en plus, cela facilite et accélère aussi grandement la phase de développement du produit.

Une grosse partie de ce travail a été le design du produit. Celui-ci devait ressembler autant que possible au véritable produit trouvé dans les avions, tout en limitant au maximum son prix.
Le faible prix a été obtenu grâce à l'utilisation d'un PC comme écran et comme cerveau du système, tandis que la ressemblance a principalement été obtenue grâce au clavier en élastomère et à la face avant peinte par sérigraphie.

Le produit pourra trouver sa place sur le marché grâce à son prix particulièrement démocratique.
Le périphérique est configurable très aisément grâce à l'application personnalisée. De plus, son firmware peut être mis à jour en quelques clics par cette même application.

Plusieurs améliorations sont envisageables dans le futur.

Le système pourrait intégrer, en plus des périphériques actuels, une souris. Lorsqu'une touche serait appuyée sur le périphérique, celui-ci simulerait un mouvement de souris vers le bouton correspondant du logiciel suivi d'un clic. Ceci permettrait de rendre le périphérique compatible avec les logiciels de simulation de FMS n'utilisant pas de raccourcis clavier.

Une autre amélioration possible est le rétroéclairage du clavier.

Enfin, il est possible de construire de nombreux nouveaux périphériques sur base de cette plateforme. Celle-ci est très polyvalente.

4. Remerciements

L'auteur tient à remercier l'ensemble du personnel de Powerdale. Il tient en particulier à remercier son promoteur Mr. J-M. Zeebergh.

5. Références bibliographiques

[1] DOERAENE Sébastien. *Créer un fichier d'aide hlp*, oct 2004, http ://sjrd.developpez.com/windows/windowstutoriels/creerhlp/

[2] MERIWEATHER Jerome. *A320 multifunction control display unit*, mar 2005 http ://www.meriweather.com/320/mcdu/mcdu.html

[3] LE MIEUX Vincent. *L'usb pour tous*. Editions techniques et scientifiques Françaises, sep 2004.

[4] USB Core team : Compaq, Intel, Microsoft, and NEC. *Universal serial bus revision 1.1 specification*, sep 1998, http ://www.usb.org/developers/docs/usbspec.zip

[5] FlyReal Team. *Flyreal*., sep 2004, http ://flyreal.free.fr/French/mcdu1_fr.htm

[6] TechScribe. *Software documentation*, dec 2004 http ://www.techscribe.co.uk

[7] WEIJERS Rob and KNOL Marcel . *Pocketfms*, jul 2003 http ://www.PocketFMS.com