REPUBLIQUE TUNISIENNE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR,
DE LA RECHERCHE SCIENTIFIQUE ET DE LA TECHNOLOGIE
UNIVERSITE DE TUNIS EL MANAR

INSTITUT SUPERIEUR D'INFORMATIQUE

RAPPORT DE PROJET DE FIN D'ETUDES

Présenté en vue de l'obtention du

Diplôme Universitaire de Technologie

Option : **Informatique Industrielle**

Par

Jaoua Mehdi

Système embarqué de transfert de fichiers USB

Entreprise / Organisme d'accueil

Centre National des Sciences et Technologies Nucléaires

Encadrant à l'entreprise : Abbass Mohamed Mehdi
Encadrant à l'ISI : Raouafi Fathi

Année Universitaire 2007-2008

REMERCIMENTS

C'est avec un réel plaisir que je réserve ces lignes de gratitude et de reconnaissance à tous ceux qui de près ou de loin, ont contribué à la réalisation de ce projet.

Je remercie infiniment Monsieur Fathi RAOUAFI, Docteur à l'ISI, pour tout son aide combien efficace et ses conseils combien pertinents. Vous étiez d'une bien vaillance de près ou de loin. Merci aussi pour avoir accepter d'être mon encadreur.

Ma gratitude et aussi exprimée à Monsieur Mohamed Mehdi ABBESS, ingénieur en chef au CNSTN, pour avoir accepter d'être mon encadreur, pour le soutien qu'il a attaché au bon déroulement de mon travail et surtout pour ces remarques et suggestions si enrichissantes.

Mes remerciements s'adresse tout particulièrement à monsieur adel trabolsi et nasir wislati……

Je ne manquerai pas de remercier tous les enseignants de l'ISI et particulièrement M. Moez FIRCHICHI, ainsi que le directeur M. Moncef TEMANI.

Qu'il me soit permis d'adresser mes remerciements à tous ceux, que j'ai oublié de citer, veuillent bien ne pas en tenir et recevoir leurs parts de remerciements.

INTRODUCTION GÉNÉRALE

L'humanité, depuis son apparition, cherche à communiquer et à s'échanger les informations afin de pouvoir évoluer dans le temps et dans l'espace. D'ou le besoin d'uniformiser cette communication pour la faciliter. Pour que deux individus dans différents endroits puissent communiquer ils doivent être sur la même longueur d'onde, autrement cette communication devient impossible avec l'interruption de l'échange des données et des informations.

Ainsi pour communiquer il est indispensable de choisir deux caractéristiques qui seront les mieux adaptés :

- La langue : doit être simple, précise et claire...
- La voie de communication : doit être simple, facile à utiliser et rapide…

Par analogie à la communication entre les Hommes dans le monde des machines, la norme USB (***Universal Serial Bus***) qui est apparue pour uniformiser la communication, non pas la communication des Hommes, mais la communication des machines.

L'USB a été conçu au milieu des années 1990 afin de remplacer les nombreux ports externes des l'ordinateurs lents et incompatibles.
Afin d'établir une communication entre ces machines, on retrouve les mêmes caractéristiques que celles choisies pour la communication entres les Hommes.

- La langue : les paquets et le protocole USB.
- La voie : 4 broches au lieu de 9 pour le port DB-9 (port série) ou encore pire 25 broches pour le DB-25 (port parallèle).

Plus précisément, je dénote, le protocole de communication USB qui devient de plus en plus adapté à toutes les machines (GSM, ordinateurs, lecteurs MP3 et MP4, caméscopes…).

Au cours de mes années d'études à l'Institut Supérieur d'Informatique (ISI), j'ai dû à maintes reprises lancer mon PC, pour copier le contenu de ma clé USB dans celle de mes collègues ou le contraire, pour leur donner une copie du cours sous forme électronique. Ainsi et au bout de deux mois, mon disque dur était plein, non pas de cours, mais de virus.

Dans ce contexte, l'objectif de ce présent Projet de Fin d'Étude (PFE) est de réaliser un outil qui permettrait d'échanger des données entre les collègues, sans avoir à scanner leurs clés à chaque copiage en évitant une grande perte de temps et en préservant mon les disques dur des virus externes.

Le principe de base est le suivant : une carte électronique ayant un microcontrôleur comme cerveau central pour gérer la communication, deux ports USB (un IN et l'autre OUT) pour brancher les deux clés (source et destination).

Le projet consiste à concevoir et réaliser une carte électronique de développement à base du µcontrôleur MICROCHIP PIC 18F452 et d'un contrôleur USB embarqué maître/esclave de CYPRESS SL811HS. L'application à développer par le biais de cette carte est la réalisation d'un système embarqué de transfert de fichiers USB.
Concevoir et réaliser un plateforme qui remplace un ordinateur lors du copiage des fichiers d'une mémoire de masse USB vers une autre, permet en effet de reproduire identiquement les bits inscris dans la première clé vers la seconde.

Ce présent rapport présentant mes travaux est scindés en deux chapitres :

- Le premier chapitre permet de présenter les différents outils pour la réalisation ce travail tel que la norme USB, les PIC, etc.

- Le deuxième chapitre a pour but de présenter les différentes étapes de réalisation de mon projet.

- Une conclusion générale clôturera ce rapport et exposera les perspectives de ce projet.

CHAPITRE 1 : PARTIE THÉORIQUE
ÉTAT DE L'ART

Introduction

Dans ce premier chapitre, je vais présenter quelques notions de base pour la réalisation de ce projet.

- Au début, je définis les PIC ou plus précisément la famille des PIC 18Fxxx.
- Ensuite je donne un bref aperçu sur la norme USB qui est bien évidement indispensable à la compréhension de ce projet.
- Enfin je vais expliquer le choix concernant le contrôleur sélectionné pour gérer la communication USB.

Une conclusion clôturera ce premier chapitre.

1. Introduction aux microcontrôleurs PIC

1.1 Définition

Un microcontrôleur est une unité de traitement de l'information de type microprocesseur à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans être obligé d'ajouter des composants annexes. Un microcontrôleur peut donc fonctionner de façon autonome après sa programmation.

En ce sens, les PIC sont particulièrement bien dotés, car ils intègrent plusieurs mémoires : de programme et de données,des ports d'entrée-sortie (numériques, analogiques, PWM, UART, bus I²C, etc.), et même une horloge même si des bases de temps externes puissent être employées. Certains modèles disposent de port et unités de traitement de l'USB.

1.2 Pourquoi le PIC 18F452 ?

En général, la famille des PIC 18f a un jeu d'instructions plus complet puisqu'il comprend de l'ordre de 75 instructions. Cette palette d'instructions étendue lui permet de faire fonctionner du code C compilé de manière nettement plus efficace que les familles précédentes. On peut les utiliser avec un quartz oscillant jusqu'à 48MHz ce qui est essentiel à la détermination de la vitesse de transmission (low-speed, full-speed ou hight-speed).

Pour accéder aux dispositifs génériques de mémoire de masse USB (clé USB), un système embarqué doit contenir un contrôleur d'hôte USB, qui peut être à part ou incorporé dans le microcontrôleur.

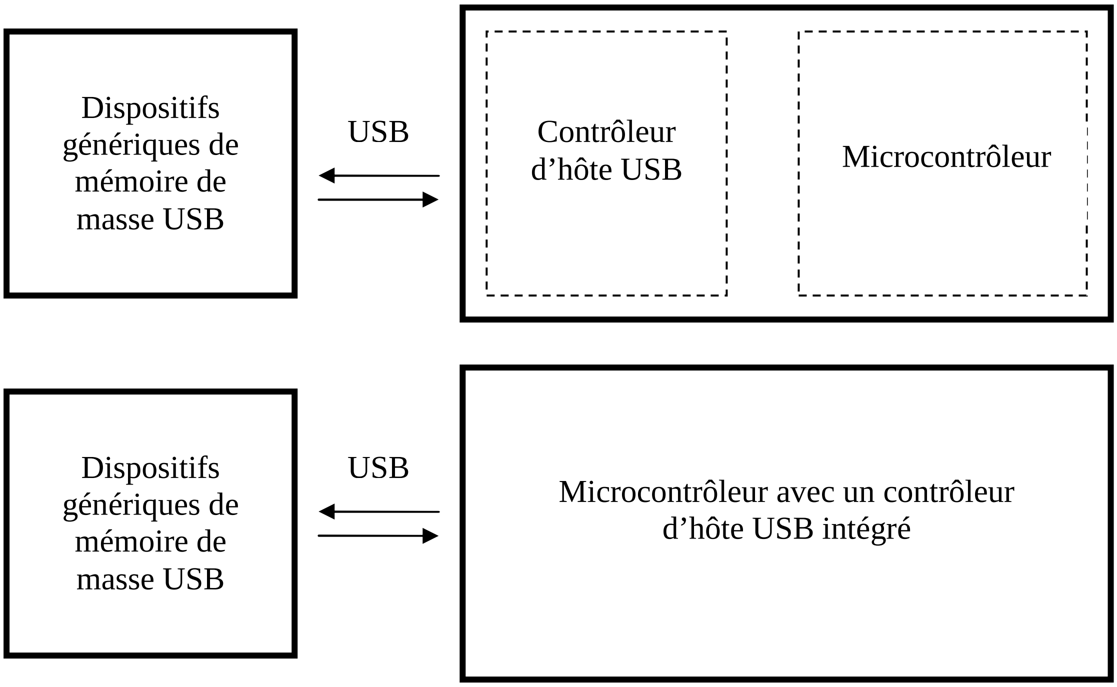

Figure 1 : Type des contrôleur USB

Pour ce travail, j'ai choisi un microcontrôleur avec un contrôleur d'hôte USB à part.
En effet, l'architecture des PIC, doté d'un contrôleur d'hôte USB intégré, est disponible sur les pics 18f2455, 18f2550, 18f4455 et 18f4550.
Cependant, la raison qui m'a poussé à ne pas choisir l'un d'eux est que leurs contrôleurs d'hôte ne sont pas capables de gérer la communication USB dans un système embarqué, c'est pour cela que j'ai choisi un microcontrôleur avec un contrôleur d'hôte USB externe.

Bien que Microchip a résolu le problème des pics 18F souffrant de leur contrôleur USB intégré, en proposant la famille des pics 24 et 32 qui jouissent de la technologie OTG (On The Go) leur permettant d'être le cerveau d'un système embarqué.
Mais je n'ai pas opté pour ces pics car vu leur création récente ils ne sont pas assez disponibles sur le marché et en plus leur coût est élevé.

2. Présentation de la norme USB

2.1 Présentation de son intérêt

L’application à développer par le biais de cette carte est la réalisation d’un dispositif permettant de transférer des fichiers via le port USB (Universel Serial Bus).

Ainsi dans cette section je propose quelques explications de la norme USB.

Pour commencer je propose d’expliciter quelques notions de base pour indispensables à la compréhension du fonctionnement du bus USB.

2.2 La norme USB

2.2.1 Présentation du port USB

L'USB, de l’anglais Universel Serial Bus comme son nom l'indique est un Bus Série. Il utilise 4 fils isolés dont 2 sont l'alimentation (+5V et GND). Les deux restants forment une paire torsadée qui véhicule les signaux de données différentiels. Il utilise un schéma d'encodage NRZI (Pas de retour à Zéro inversé) pour envoyer des données avec un champ «*sync*» de manière à synchroniser les horloges de l'Hôte (le PIC 18F452) et le récepteur (les clés USB).

L'USB version 1.1 comprenait 2 vitesses, un mode vitesse rapide de 12Mbits/s et un mode vitesse lente de 1,5Mbits/s. Le mode de 1,5Mbits/s est plus lent et moins sujet aux perturbations Electro-magnétiques (EMI) réduisant ainsi le coût de pertes de ferrites et des composants de qualité. Par exemple les Quartz peuvent être remplacés par des résonateurs meilleur marchés.

L'USB 2.0 qui est encore à la veille de voir le jour sur les ordinateurs grand public a fait monter les enchères jusqu'à 480Mbits/s. Le 480Mbits/s est connu sous le nom de mode Haute vitesse et a été crée pour entrer en compétition avec le Bus Série Firewire.

Les vitesses USB sont donc :

- Vitesse Haute - 480Mbits/s.....*High Speed*
- Vitesse Pleine - 12Mbits/s.....*Full Speed*
- Vitesse Basse - 1,5Mbits/s.....*Low Speed*

Le Bus Série Universel est contrôlé par l'Hôte. Il ne peut y avoir qu'un Hôte par Bus.

L'USB supporte le système «plug & play» branchement à chaud avec des drivers qui sont directement chargeable et déchargeable. L'utilisateur branche simplement l'appareil sur le Bus. L'hôte détectera cet ajout, interrogera l'appareil nouvellement inséré et chargera le driver approprié. L'utilisateur final n'a pas besoin de se soucier des terminaisons, de termes tel que les adresses des ports, ou de la réinitialisation de l'ordinateur. Une fois que l'utilisateur a terminé, il peut simplement retirer le câble, l'hôte détectera cette absence et déchargera automatiquement le driver.

Une autre caractéristique intéressante de l'USB réside dans ces modes de transferts. L'USB soutient des transferts de contrôles, d'interruptions, en Bloc et Isochrone. Ce dernier, permet à un appareil de réserver une approximation définie de la bande passante avec un temps d'attente garanti.

2.2.2 Connecteurs et câbles USB

On utilise des couleurs standard pour les fils intérieurs des câbles USB de façon à faciliter l'identification des fils d'un constructeur à un autre.

Numéro de la broche	**Couleurs des câbles**	**Fonction**
1	Rouge	Vbus (5 V)
2	Blanc	D_-
3	Vert	D_+
4	Noir	Masse

Figure 2 : Désignation des câbles de l'USB

2.2.3 Description de la transmission USB

L'USB utilise une paire de transmission différentielle pour les données. Celle-ci étant codé en utilisant le NRZI et est garni de bits pour assurer les transitions adéquates dans le flot de données. Sur les appareils à vitesse basse et pleine un '1' différentiel est transmis en mettant D+ au dessus de 2,8V grâce à une résistance de 15k ohms relié à la masse et D- en dessous de 0,3V avec une résistance de 1,5kΩ relié à 3,6V. D'autre part un différentiel '0' correspond à D- plus grand que 2,8V et D+ inférieur à 0,3V avec les mêmes résistances de rappel état haut/bas adéquates. Le récepteur défini un différentiel '1' avec D+ plus grand de 200 mV que D- et un différentiel '0' avec D+ plus petit de 200 mV que D-. La polarité du signal est inversée en fonction de la vitesse du BUS. En conséquence les états référencés par les termes 'J' et 'K' sont utilisés pour signifier les niveaux logiques. En vitesse basse, un état 'J' est un différentiel '0'. En vitesse haute, un état 'J' est un différentiel '1'. Le BUS basse et pleine vitesse a une impédance caractéristique de 90Ω +/-15%. Il est donc important d'observer la documentation technique lorsque on sélectionne les résistances, les caractéristiques électriques séries pour D+ et D- afin d'équilibrer l'impédance.

2.2.4 La reconnaissance des périphériques : énumération du bus

Quand un périphérique USB est raccordé ou débranché du bus USB, le système hôte utilise un processus appelé « énumération » pour identifier et gérer le statut du périphérique dont l'état a changé. Quand un périphérique est rattaché (et est sous tension pour les autoalimentés) au bus USB les actions suivantes sont démarrées :

1. La carte auquel est rattaché le périphérique informe l'hôte de cet événement. A ce moment là, le périphérique est dans l'état sous-tension et le port auquel il est rattaché est dévalidé.
2. L'hôte détermine la nature du changement en interrogeant le µcontrôleur.
3. Maintenant que l'hôte connaît le port de rattachement auquel le nouveau périphérique est rattaché, il attend 100 mS pour permettre au processus d'alimentation et d'initialisation de celui-ci de se terminer. Il valide alors le port de rattachement et envoi un « RESET » à ce port.
4. Le µcontrôleur maintient le signal de « RESET » sur le port durant 10mS, et quand celui-ci se termine le port est validé. Le périphérique USB est maintenant dans l'état de défaut et ne doit pas utiliser plus de 100 mA sur le bus USB. Tous ses registres sont réinitialisés et il répond à l'adresse par défaut.
5. L'hôte lui assigne une adresse unique et le faisant passer dans l'état adressé.
6. Avant que le périphérique reçoive son adresse il est accessible par l'adresse par défaut. L'hôte lit sa description et détermine les besoins en mémoire nécessaire à ce périphérique.
7. L'hôte lit toutes les configurations possibles du périphérique, ce processus peut durer plusieurs mS.

L'hôte choisi une configuration parmi celles disponibles du périphérique. Celui-ci passe dans l'état configuré et toutes les données de configuration sont celle de la configuration choisie. Le périphérique peut maintenant consommer le courant qui lui est nécessaire sur le bus. Il est maintenant prêt à fonctionner. Quand un périphérique USB est déconnecté, le µcontrôleur en envoi la notification à l'hôte. Le port auquel il était attaché est dévalidé. Après avoir reçu la notification de détachement l'hôte actualise ses informations concernant la topologie du bus.

2.2.5 Le stockage de masse et la clé USB

2.2.5.1 Présentation du stockage de masse

Dans le domaine informatique le premier besoin qui est apparu c'est le besoin de sauvegarder des données. Pour cela de nombreux périphériques ont été développés : disques durs, cartes mémoire, clés USB. Il a donc fallu choisir arbitrairement une unité de stockage élémentaire. L'élément qui est utilisé sur les périphériques de stockage de masse est appelé un « secteur », il est composé de 512 octets soit 4096 bits. Ce choix résulte d'un compromis entre la taille des adresses et la taille des données.

Du point de vue de l'utilisateur un périphérique de stockage de masse apparaît comme une succession de secteurs. On retrouve cette organisation en secteur sur tous les périphériques de stockage de masse courants. Cette organisation implique une écriture/lecture au minimum d'un élément mémoire, c'est à dire au minimum une écriture/lecture de 512 octets. Un secteur est indivisible, par conséquent lorsqu'on voudra modifier un octet sur une clé USB, il faudra commencer par lire le secteur qui contient l'octet, modifier l'octet, puis réécrire entièrement le secteur.

2.2.5.2 Les fonctions spécifiques aux clés USB

Une clé USB est une mémoire flash qui fonctionne sur le bus USB, elle répond donc au protocole USB vu dans le chapitre précédent.

Dans un premier temps la clé répondra aux requêtes USB standards (énumération) puis elle répondra aux requêtes spécifiques qui permettront d'écrire et de lire dans sa mémoire flash.

L'USB comme nous l'avons vu lors de l'énumération est un protocole qui permet de diviser les périphériques en différentes classes. Cette division permet aux utilisateurs de ne pas avoir à installer un driver sur tous les ordinateurs sur lesquels ils branchent une clé USB.

L'ordinateur va interroger la clé pour savoir à quelle classe elle appartient. Il utilisera ensuite un driver générique à la classe : la généricité est l'un des grands avantages de l'USB !!!

Les clés USB comme les disques durs, les appareils photos, appartiennent à la classe "stockage de masse" (Mass Storage Class).

Le numéro de cette classe est donné dans le descripteur d'interface, il s'agit du numéro 0x08.

Cette classe se subdivise en deux sous classes :

- Bulk only
- Bulk/Control Transfer/Transfert Interrupt

Ces deux sous classes définissent deux protocoles différents d'accès au stockage des données : le premier utilise uniquement les transferts **Bulk** pour échanger des données, tandis que le second utilise le **Control Transfert** pour envoyer la commande, le **Bulk** pour envoyer les données et le **Transfert Interrupt** pour recevoir l'acquittement de la commande.

Les PCs utilisent uniquement le **Bulk Only** qui est un protocole plus rapide et qui peut être considéré plus simple. Ce protocole utilise les fonctions SCSI génériques qu'il empaquette dans des paquets USB. Ce protocole est donc beaucoup plus générique que son homologue. D'autre part le protocole Bulk/Control Transfer/Transfert Interrupt est condamné à disparaître dans les années à venir.

Le protocole Bulk Only :

Le protocole Bulk Only utilise uniquement 2 types de paquet USB :

- Des paquets OUT
- Des paquets IN

Ce protocole s'organise en trois phases :

1. Envoi de la commande : CBW
2. Envoi ou réception des données
3. Réception d'acquittement de la commande : CSW

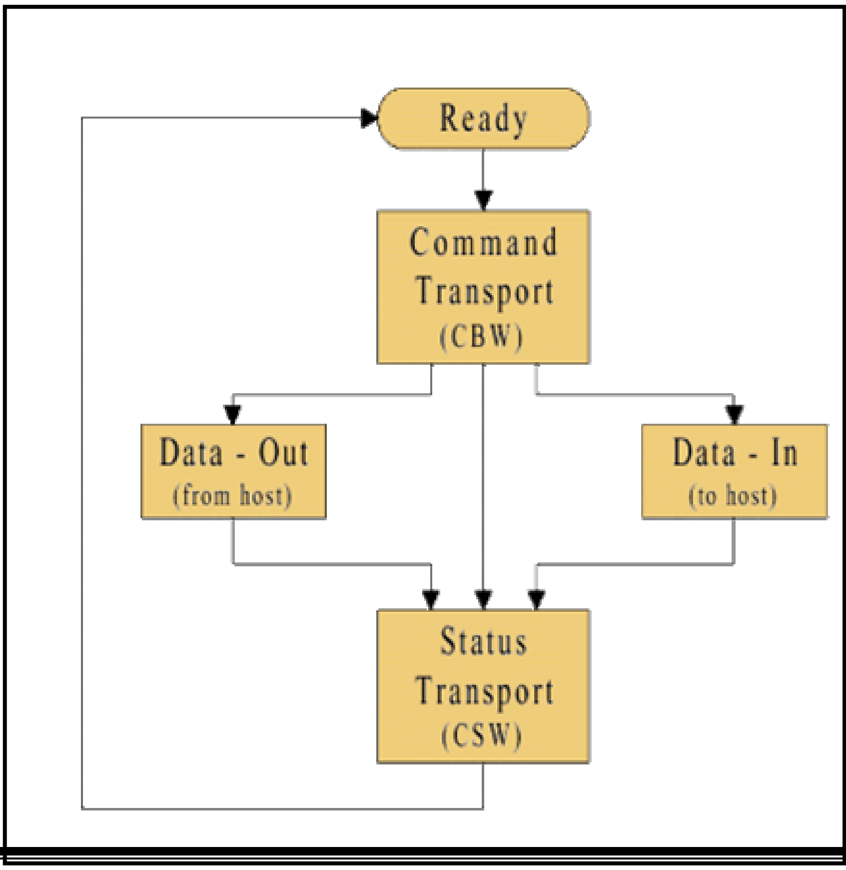

Figure 3 : Illustration du transfert Bulk

<u>Le paquet CBW :</u>

C'est le paquet dans lequel la commande est envoyée, il est composé de 31 octets envoyés dans un paquet OUT.

Les quatre premiers octets sont la signature du paquet 'USBC'. Cette signature reste constante et permet la reconnaissance d'un paquet CBW.

Les quatre octets suivants sont un 'tag' qui sera retourné avec le paquet CSW. Ces quatre octets ne sont pas utiles pour le fonctionnement de la clé, cependant le host pourra lors de la réception du paquet CSW contrôler que le 'tag' est identique à celui envoyé dans la commande CBW.

On retrouvera ensuite une série d'octet qui coderont la commande à réaliser, par exemple read (address, sector number,...).

<u>Le paquet de données :</u>

C'est le paquet qui contient les données utiles, c'est à l'intérieur de ce paquet que l'on écrira les données.

<u>Le paquet CSW :</u>

C'est le paquet d'acquittement de la commande. Ce paquet renvoie la/les erreur(s) qui se sont produites lors de la précédente commande.

2.2.6 Le protocole USB

Contrairement à la RS232 et des interfaces sérielles similaires ou le format des données envoyées n'est pas défini, l'USB est composé de plusieurs couches de protocoles. Bien que cela semble compliqué, la plupart des circuits intégrés contrôleur d'USB s'occuperont de la couche inférieure, la rendant ainsi presque invisible au regard du concepteur final.

Chaque transaction USB consiste d'un :

- Paquet Jeton (Token) (en tête définissant ce qu'il attend par suite)

8 bits	8 bits	7 bits	4 bits	5 bits
SYNC	PID	ADDRESS	ENDP	CRC

Figure 4 : Trame du paquet jeton

- Paquet DATA optionnel (contenant la « charge utile » (playload))

8 bits	8 bits	0 to 512 bits	16 bits
SYNC	PID	PAYLOAD	CRC

Figure 5 : Trame du paquet data

- Paquet d'Etat (utilisé pour valider les transactions et pour fournir des moyens de corrections d'erreurs).

Le bus USB est un bus géré par l'hôte, ce dernier initie toutes les transactions. Le premier paquet, aussi appelé Jeton est produit par l'hôte pour décrire ce qui va suivre et si la transaction sera en lecture ou écriture et ce que sera l'adresse de l'appareil et la terminaison désignée. Le paquet suivant est généralement un paquet de données transportant la " charge utile " et est suivi par un paquet " poignée de mains " (handShaking), signalant si les données ou le jeton ont été reçus correctement ou si la terminaison est bloquée, ou n'est pas disponible pour accepter de données.

3. <u>Présentation du controleur d'hote usb SL811HS</u>

3.1 Présentation générale

Le SL811HS est un contrôleur incorporé d'USB Host/Slave capable de communiquer avec les périphériques à toute vitesse ou à vitesse réduite.

Le SL811HS peut être connecté aux dispositifs tels que des microprocesseurs, microcontrôleurs, DSPs, ou directement à une variété de bus tels que le ISA, PCMCIA, et d'autres.

Le contrôleur USB de SL811HS se conforme aux spécifications 1.1 de l'USB.

Le contrôleur d'USB maitre/esclave de SL811HS incorpore la fonctionnalité d'interface série de l'USB avec les émetteurs récepteurs internes de full-/low-speed.

Le SL811HS fonctionne en mode toute vitesse d'USB à 12 Mbps, ou au mode 1.5-Mbps à vitesse réduite.

L'interface du microprocesseur et du port des données de l'SL811HS fournit une circulation de données de 8 bits I/O avec l'appui d'interruption.

3.2 Diagramme de bloque

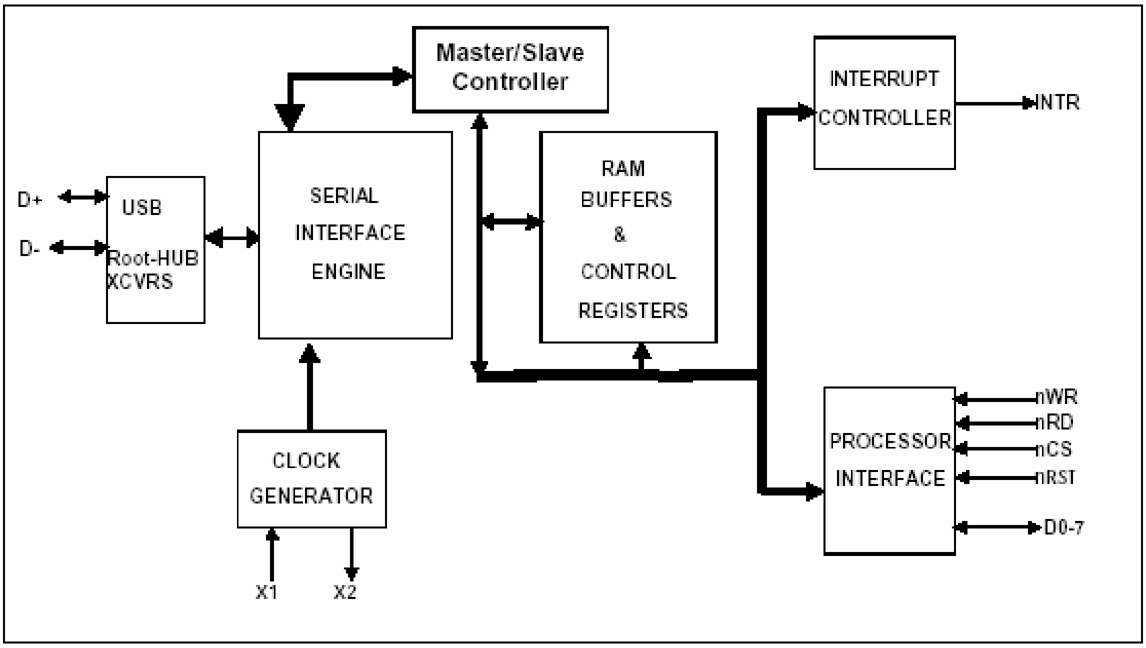

Figure 6 : Illustration du bloque diagramme de l'SL811HS

Conclusion

Au cours de ce chapitre j'ai présenté les différentes parties nécessaires pour la réalisation de mon projet.

La conception et la réalisation d'une carte de transfert de fichier USB, constituent une application intéressante dans la mesure où elle intègre de nouvelles techniques tel que l'introduction des µcontrôleurs, la programmation In Situ etc.

Ainsi, dans le chapitre suivant, je vais présenter les différentes étapes de réalisation après avoir clarifié les points ambigus, vu que j'ai essayé de les expliquer dans la partie théorique.

CHAPITRE 2 : PARTIE PRATIQUE : RÉALISATION

Introduction

Dans ce chapitre, je vais exposer les différentes étapes de réalisation de mon projet qui consiste à développer un système embarqué permettant le transfert de fichier d'une clé vers une autre.

Il est vrai que, vu d'un angle purement théorique, la conception d'un tel système n'est pas si difficile à concevoir cependant la réalisation pratique a révélée tout à fait le contraire.

Étant donné la complexité et les contraintes techniques de la réalisation pratique mon présent travail se concentrera sur la réalisation d'une carte électronique permettant la communication avec une seule clé USB en suite par duplication de ce travail il serait possible d'assurer le transfert de fichier entre deux clés USB.

1. Technique de réalisation d'un circuit imprimé

1.1 Présentation de son intérêt

La première étape de mon projet, porte sur l'apprentissage des principes de conception et de réalisation d'un circuit imprimé.

Avant de passer à l'impression de la carte, il faut avoir le schéma du circuit, appelé typon.

Le schéma du circuit (typon) en question ne peut être conçu qu'après avoir accompli la conception de la carte avec un logiciel tel que le PROTEUS 7 ISIS.

J'expose ainsi la procédure permettant de bien mener cette phase d'impression.

1.2 Étapes d'impression du circuit

Ce paragraphe comportera plusieurs étapes dont, la confection du typon, la préparation de la plaque, l'insolation, la révélation, la gravure, le nettoyage, l'étamage, le perçage, le montage, le nettoyage de la résine et enfin comme toute œuvre qui se respecte, le vernissage.

1.2.1 Matériels nécessaire

- Plaque cuivrée simple face photosensible
- Révélateur
- Perchlorure de fer (chlorure ferrique)
- Alcool à brûler
- Insoleuse
- Divers bacs
- Gants de protection

1.2.2 Le typon

Il peut être conçu à partir de la photocopie d'un typon proposé par une revue ou sur le web pour une réalisation existante ou réalisée à l'aide d'un logiciel pour une réalisation personnelle par exemple.

Il faut imprimer deux transparents pour photocopieuse à l'aide d'une imprimante laser ou jet d'encre ou encore sur une photocopieuse. Cette duplication est importante car l'opacité des encres est souvent trop pâle et il risquerait de se produire des micros coupures. Les deux transparents seront superposés parfaitement et agrafés à leurs extrémités.

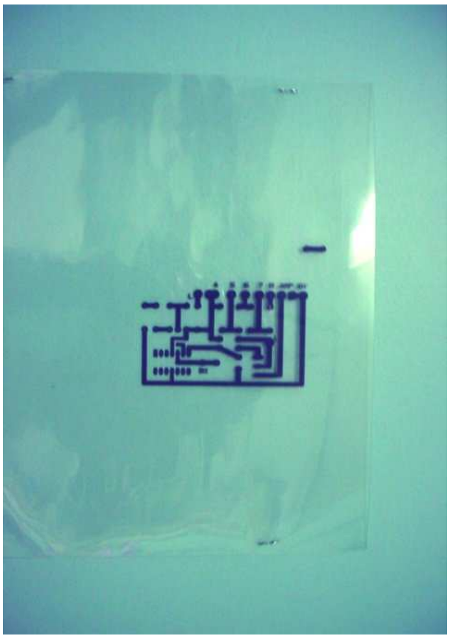

Figure 7 : Les deux transparents superposés

1.2.3 L'insolation

Les plaques photosensibles sont revêtues d'un papier protecteur opaque qui protège la face photosensible de l'ultra violet pendant son stockage. On dispose le typon sur la glace de l'insoleuse, **attention au sens de positionnement**, ensuite on retire le papier de protection de la plaque, puis on présente la face cuivrée photosensible sur le typon, puis l'on referme l'insoleuse.

Figure 8 : Insoleuse avec 4 tubes à UV

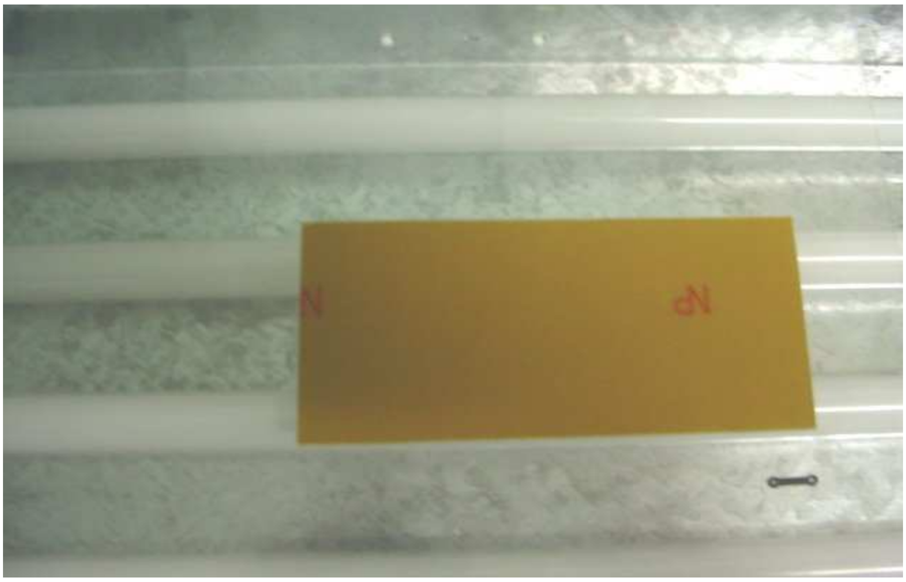

Figure 9 : Le typon est sous la plaque photosensible

Ensuite on allume l'insoleuse, le temps d'insolation est variable, mais 2mn30 est un bon compromis.

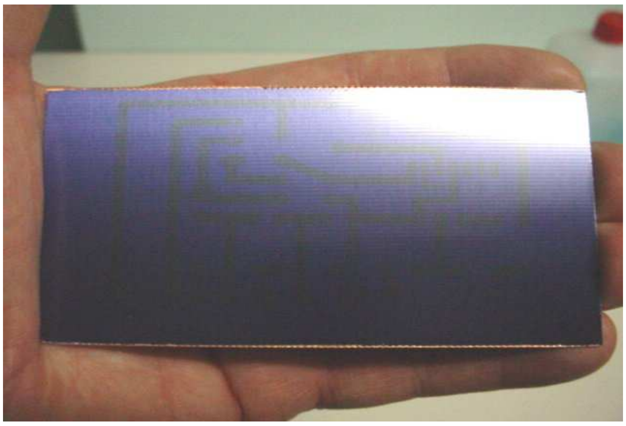

Figure 10 : Apparition du dessin après l'insolation

1.2.4 La révélation

La plaque sortie de l'insoleuse présente le dessin du circuit déjà visible. Il faut plonger cette plaque dans le produit révélateur jusqu'à ce que le dessin apparaisse parfaitement. Sortir la plaque et la rincer sous le robinet. A chaque manipulation, tenir la plaque par les côtés en prenant bien garde à ne pas mettre de trace de doigt. Votre plaque est alors prête pour la gravure.

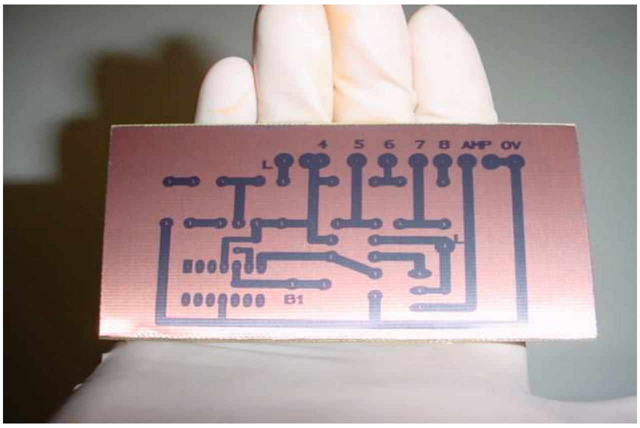

Figure 11 : La plaque après révélation

1.2.5 La gravure

Placer la plaque révélée dans le perchlorure de fer face cuivrée vers le haut puis agiter doucement le bac qui flotte de manière à ce que le circuit se promène d'un bord à l'autre. Le temps de gravure sera fonction de divers paramètres : Température du bain, activité du perchlorure de fer, surface du cuivre à dissoudre, épaisseur du cuivre etc...

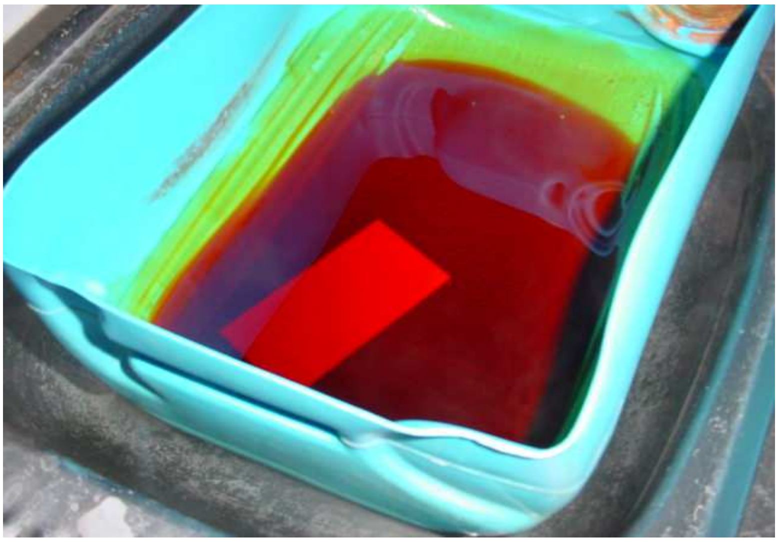

Figure 12 : Plaque entrain d’être gravée dans le perchlorure de fer

Le cuivre sera dissous partout où il n'y à pas de pistes, bien veiller à ce que tout soit bien parfaitement "grignoté" entre chaque piste et qu'il ne reste pas de court-circuit intempestif avant de le sortir. Ne le laissez pas trop longtemps, surtout si certaines pistes sont fines.

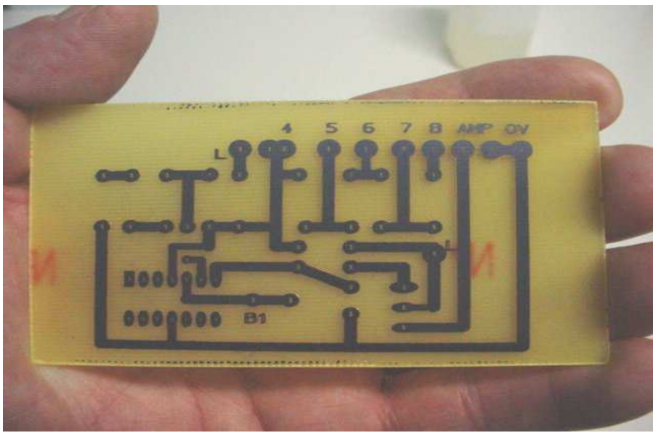

Figure 13 : La plaque après gravure

Le produit de protection des pistes à l'aide d'un chiffon imbibé d'alcool à brûler jusqu'à ce que les pistes soient d'une belle couleur cuivrée.

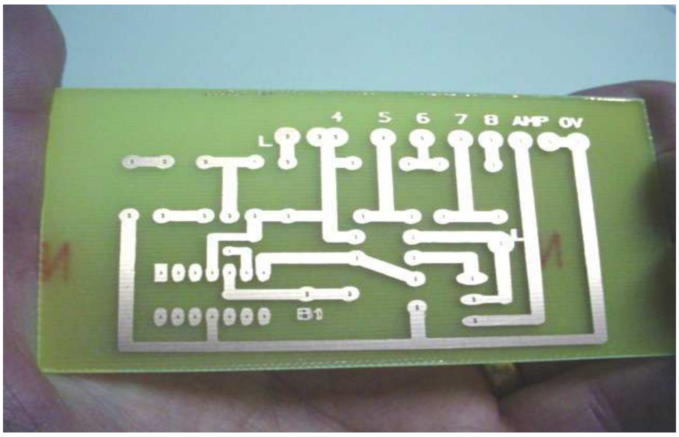

Figure 14 : La plaque après nettoyage à l'alcool à brûler

1.2.6 Le perçage

Pour la plupart des composants classiques on utilisera un forêt de 0,7mm, pour les plus gros 0,8mm et 1mm. Il est souhaitable de tourner très vite afin d'obtenir une bonne vitesse de coupe. Mais rassurez vous avec un bon forêt à bonne vitesse on peut percer plusieurs milliers de trous, de quoi faire quelques circuits.

Figure 15 : Perçage de la plaque

1.2.7 Le montage

On mettra en place les composants en commençant par les plus petits, en veillant au sens pour ceux qui sont polarisés. On effectuera les soudures à l'aide d'un fer si possible thermo staté, puis on coupera les pattes des composants à ras de la soudure (attention aux yeux, ça saute), maintenir la patte avec un doigt pendant la coupe.

Visuellement une mauvaise soudure se reconnaît assez facilement, elle a l'apparence d'une **boule** alors qu'une bonne soudure a l'apparence d'un **volcan.**

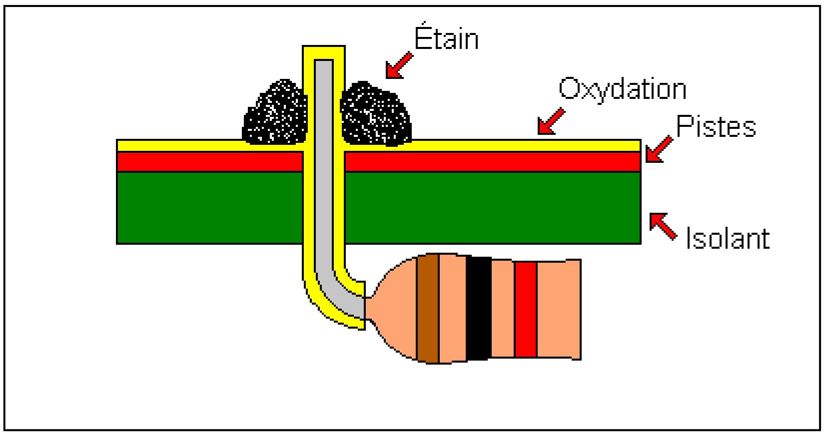

Figure 16 : Soudure en boule

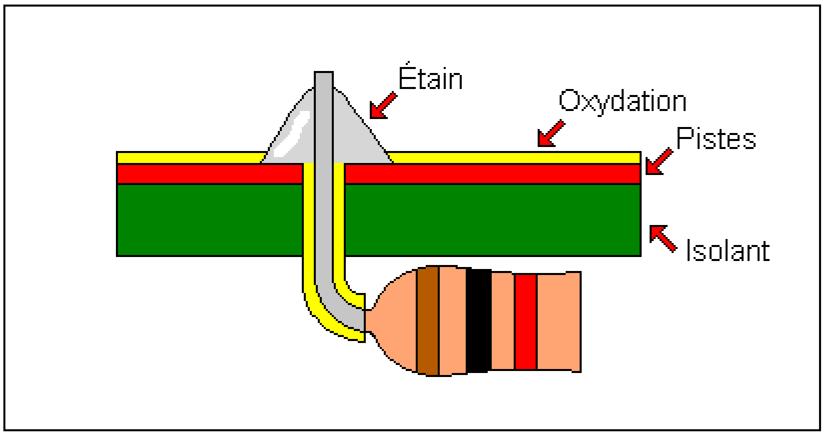

Figure 17 : Soudure en volcan

2. Le programmateur universel

2.1 Présentation du fonctionnement du programmateur universel

La simplicité du montage est dû au mode de programmation série.

Seuls 3 fils, sans compter l'alimentation, sont nécessaire pour programmer mon microcontrôleur PIC :

- ✓ DATA : délivre les données au niveau logique TTL. Ce signal est issu en fait de la broche DTR (Data Terminal Ready) du port série ;
- ✓ CLK synchronise l'envoi des données vers le composant à programmer (ou à lire). Ce signal provient de la broche RTS (Ready To Send) du port série ;
- ✓ Vpp est la tension de programmation. C'est la broche Tx du port série, qui permet par l'intermédiaire des transistors T1 et T2 d'obtenir à la demande une tension de 13 V.

Dans le cas des PIC, le passage de la patte MCLR de 0 à 13 V (± 0.5 V) est la condition sine qua non pour déclencher le processus de programmation dit à «haute tension».
En même temps les pattes RB6 et RB7 doivent être maintenues au niveau bas, et sur certains circuits la patte RB3 également (pour éviter l'entrée en mode programmation «basse tension»).
Le processus de programmation «haute tension» ICSP (In Circuit Serial Programming) ne peut être inhibé, contrairement au mode «basse tension» LVP (low voltage programming) qui se contente d'une tension de 5 V, mais qui peut être bloqué par le mot configuration.
La tension de 13 V est obtenue par le régulateur IC1, en intercalant les diodes D2 et D3 sur la patte de réglage de la tension de référence. Puis vient le transistor T 2, lui-même piloté par le transistor T1 qui est commandé par la broche Tx du port série.
Les signaux de données et d'horloge sont d'abord adaptés au niveau logique TTL par le MAX 232.Les portes NAND qui suivent permettent de les inverser avant de les acheminer vers les supports des PIC. Les données arrivent toujours sur la patte RB7, et l'horloge, sur la patte RB6.
Une led rouge a été connectée sur la tension de programmation pour indiquer l'activité du programmateur. Tant que celle-ci sera allumée, le composant à programmer ne devra pas être touché.

2.2 Réalisation programmateur universel

2.2.1 Conception du programmateur

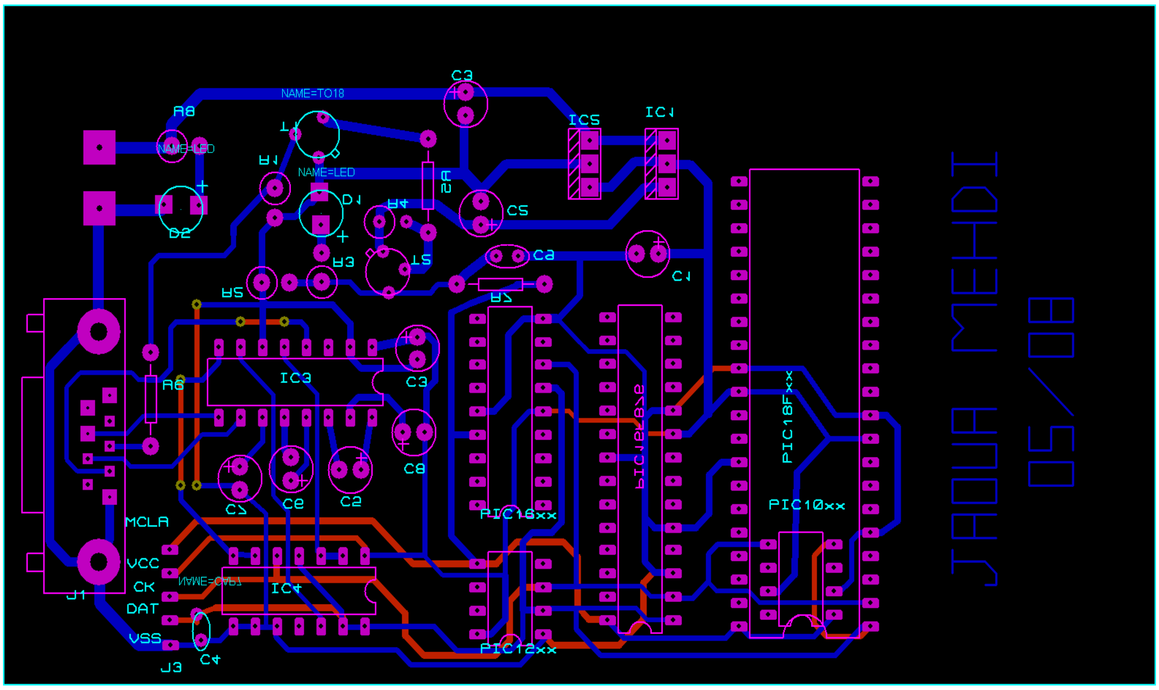

Figure 18 : Illustration du circuit à imprimer

Dans cette figure je présente la disposition des différents composants nécessaire à la mise en œuvre de mon programmateur universel.

J'ai imprimé ce circuit tout en respectant les étapes de réalisation d'un circuit imprimé (voir paragraphe 1.2 du chap 2).

2.2.2 Liste et referens des composants

Résistances :

R1, R4 :100 kΩ

R2 : 47 kΩ

R3, R8 : 2,2 kΩ

R5 : 4.7 kΩ

R6 : 10 kΩ

R7 : 1 kΩ

Condensateurs

C1, C2 : µF

C3 : 100 nF

C4 : 100 pF

C5, C6, C7, C8 : 1 µF

C9 : 22nF

Semi-conducteurs :

IC1 : régulateur 7808

IC2 : régulateur 7805

IC3 : MAX 232

IC4 : CD4039

D1 : led rouge « voyant de programmation »

D2 : led verte « voyant sous tension »

T1 : transistor NPN BC547

T2 : transistor PNP BC 557

Connecteurs et supports :

J1 : enbase DB9 femelle coudée pour circuit imprimé

1 support 40 pattes (large)

1 support 28 pattes (étroit)

1 support 18 pattes (étroit)

1 support 16 pattes (étroit)

1 support 14 pattes (étroit)

2 support 8 pattes (étroit)

2.2.3 Configuration de ICPROG

Après avoir terminer la mise en œuvre du programmateur, je vous expose la façon d'utiliser ce programmateur.

Pour pouvoir utiliser le programmateur il faut avoir un logiciel de programmation des PIC. En ce qui me concerne j'ai fais le choix d'utiliser ICPROG, qui permet une programmation facile, spécifique et efficace de mon PIC.

A ce stade je vais présenter les différentes étapes nécessaires à la configuration de ICPROG.

Après avoir exécuté le programme d'installation, il est tout d'abord nécessaire de régler la configuration «Hardware» du logiciel (figure 16) pour cela, il faut aller dans le menu «configuration → configuration hardware» et régler le type de programmateur sur «JDM programmer». Il faut bien s'assurer que le port série sélectionné correspond à celui sur lequel on a branché le montage.

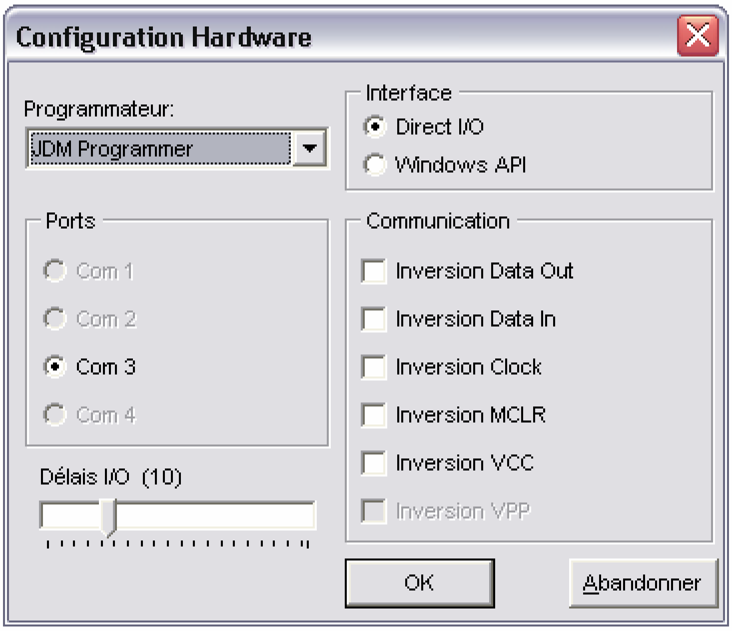

Figure 19 : Configuration Hardware de ICPROG

La temporisation «Delay I/O» pourra être réglée par défaut à 10.

Comme j'utilise le système d'exploitation Windows XP, j'ai dû modifier le mode d'accès au port COM en validant la case «Windows API».

Place maintenant à la configuration du driver. Pour cela il faut aller dans le menu « configuration → options → misc» et cocher l'option «Active driver NT/2000» (figure 17)

Une fois la configuration de ICPROG est terminée. Je vais expliciter la manière d'utiliser ce logiciel.

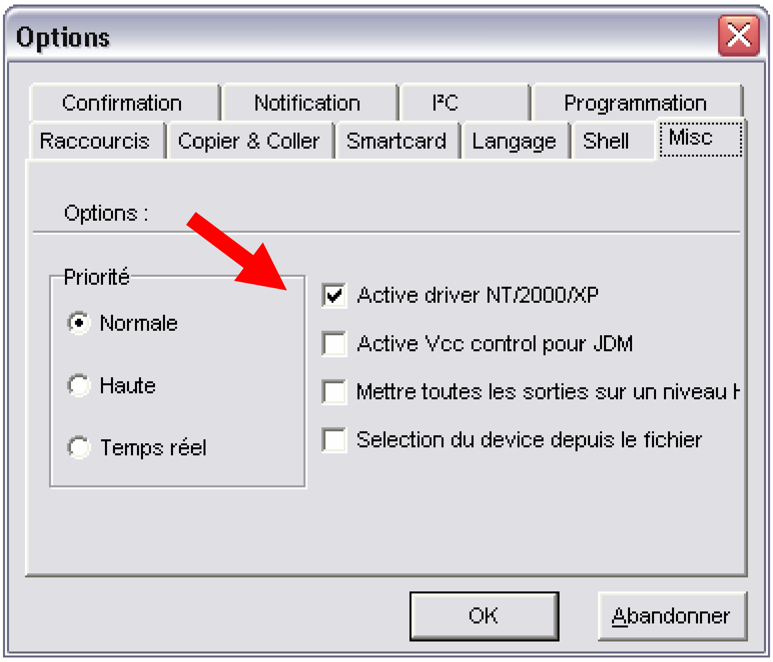

Figure 20 : Configuration Software de ICPROG

2.2.4 Utilisation

A partir du moment où il n'y a plus d'erreur de câblage, où ICPROG est configuré, et le câble série est branché sur le port COM correspondant, le montage doit immédiatement fonctionner.

Dans tout les cas la led rouge doit s'allumer au moins de façon brève, sinon il y a problème du coté «Hard».

Si tout fonctionne correctement, il faut placer le PIC sur le support correspondant en faisant attention au sens (indiquer par le demi cercle), puis charger le fichier hexadécimal (.HEX) et terminer par le lancement du chargement dans la mémoire du PIC.

3. <u>Réalisation de la carte de transfert de fichier USB</u>

3.1 Présentation de la conception de l'architecture

Dans ce chapitre, je vais expliciter la manière par laquelle j'ai procédé pour concevoir cette carte de transfert de fichier.

Étant donné la complexité de faire communiquer une clé USB avec un microcontrôleur, je me suis contenté à faire communiquer une seule clé USB avec mon microcontrôleur.
Une fois cette communication est réalisée il serait possible de dupliquer ma carte et ainsi aboutir au transfert de fichiers désiré.

La figure suivante expliquera ce principe.

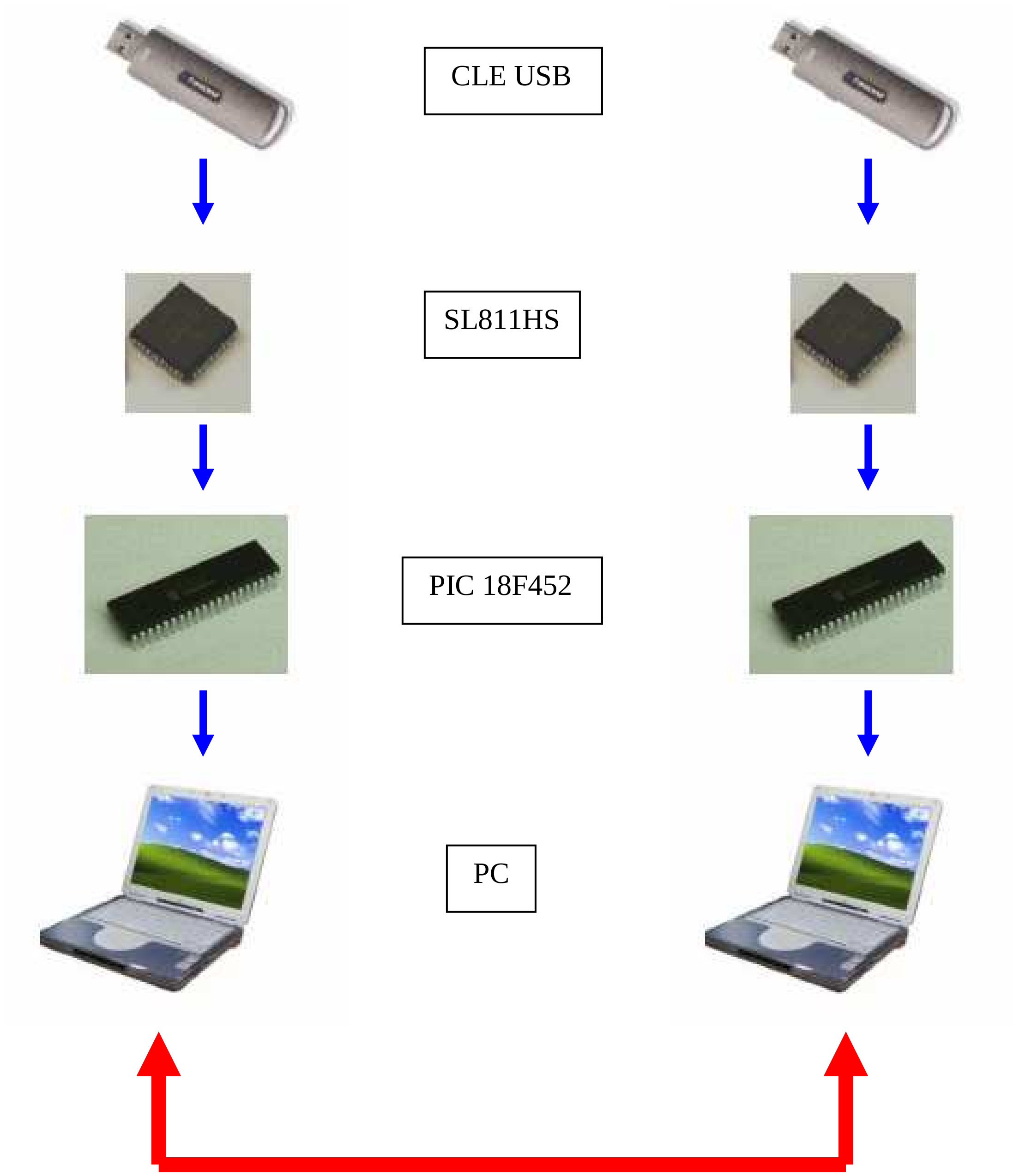

Figure 21 : Architecture étendu de la communication entre deux clés USB

Le cadre bleu de la figure 22 montre la chaîne permettant la communication entre une clé USB et un terminal de visualisation (ici un ordinateur). Par contre, la totalité de la figure 21 montre, par symétrie, comment faire pour communiquer 2 clés USB.

Ainsi dans un premier temps je m'intéresse uniquement à la partie encadrée en bleu. Et pour que ces 2 clés puissent communiqué, il suffit de réduire cette architecture comme vous la montre la figure 22.

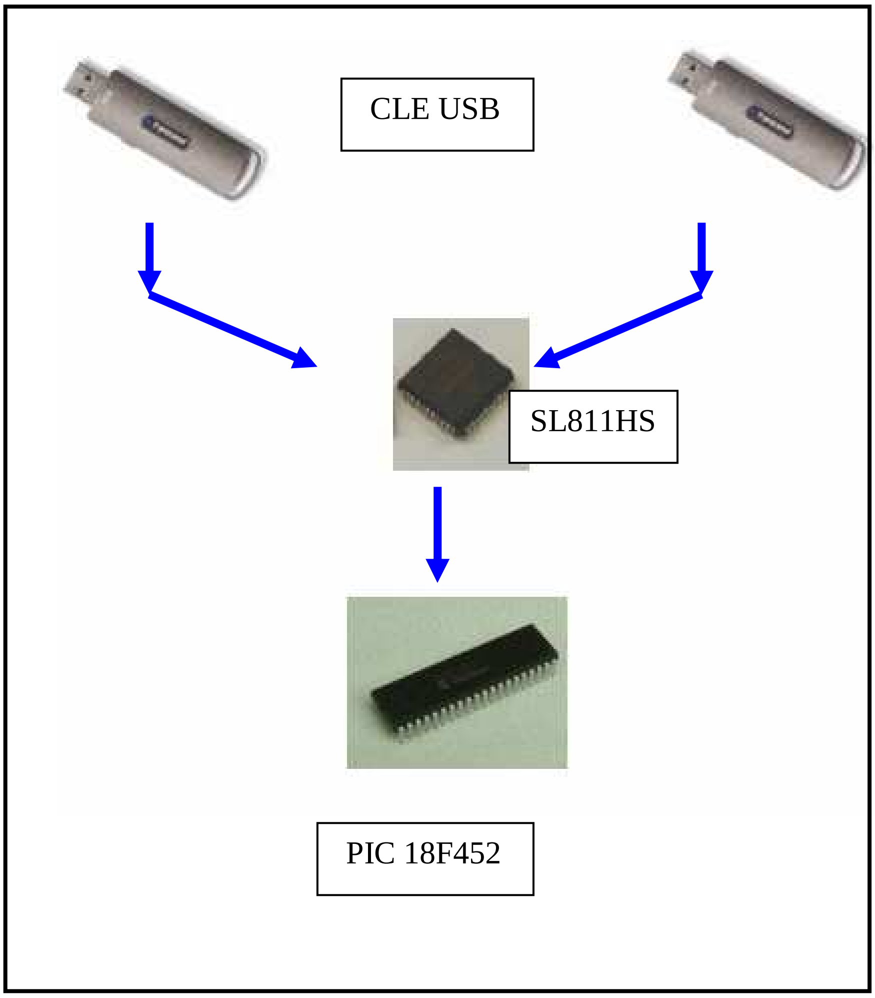

Figure 22 : Architecture simplifiée de la communication entre deux clés USB

La figure 22, montre l’architecture que j’ai conçu et qui permettra le transfert de fichier. Cependant il faut implémenté un code correspondant, incluant la logique du HUB USB, ou ajouté un DEMUX 2 vers 4 à la sortie de D+ et D- de l’SL811HS (voir figure 23).

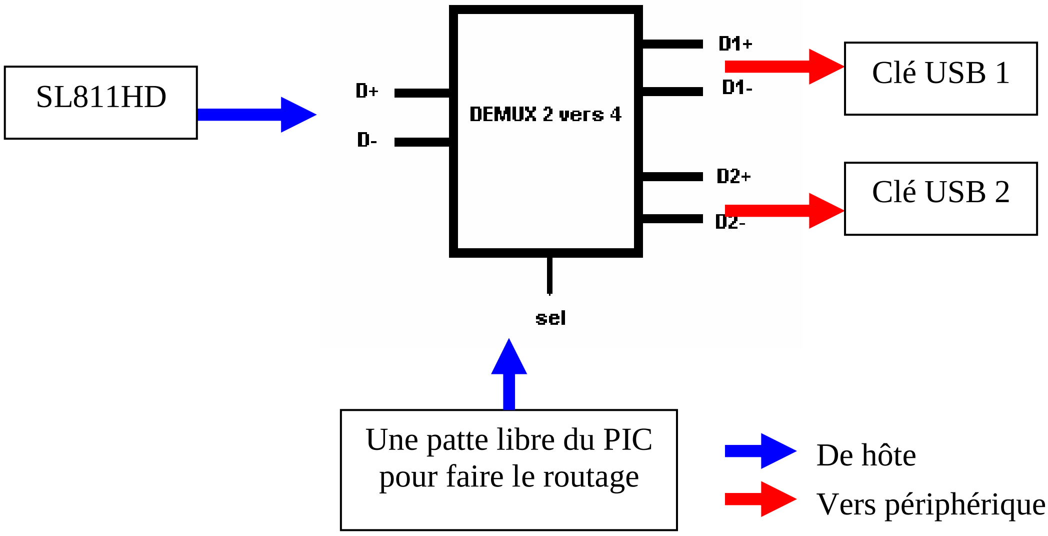

Figure 23 : Interconnexions de l'hôte avec les périphériques

3.2 Architecture de la carte

3.2.1 Conception

Avant de se lancer dans la programmation il faut commencer par réaliser une carte fonctionnelle qui contient le nécessaire pour réaliser un host USB; un microcontrôleur et un SEI. Un SEI (Serial Engine Interface) est l'interface matérielle entre le microcontrôleur et le bus USB, cette interface gère toute la partie matérielle du bus : adaptateur de niveau, décodage, bit stuffing, stockage dans les buffers. Ce composant est appelé plus généralement PHY (comme physique).
Ici j'ai décidé de mettre en oeuvre le composant Cypress le SL811HS (voir figure 3). Ce composant est un host/device qui peut travailler en FULL SPEED ou en LOW SPEED. Ici je ne travaillerai qu'en FULL SPEED ; les clés LOW SPEED seront rejetées. Les clés actuelles travaillent en FULL SPEED et pour la majorité en HIGH SPEED. A noter que les clés que l'on appelle couramment USB2.0 sont généralement des clés travaillant en HIGH SPEED, ces clés seront supportées et travailleront à vitesse réduite (12 Mbits).

Ce composant est directement connecté au microcontrôleur à l'aide une interface parallèle 8 bits et quelques signaux de contrôle. Son alimentation doit être une alimentation en 3.3V ce qui implique l'utilisation d'un régulateur ici réalisé avec un transistor et une diode zener (Q3 et D4). A noter que le microcontrôleur est lui alimenté en 5 V, mais il n'y a pas de problème en ce qui concerne la connexion au SL811 car celui ci est "5V tolérant".

On retrouve un port série sur la carte qui permet de "débuguer au printf()". Le port USB de type A reçoit les 2 fils de donnée du bus USB et son alimentation. L'alimentation de celui-ci est permanente ce qui implique que le périphérique sera mis sous tension dès son branchement, de plus il n'y aura pas de contrôle de surintensité. Il est utile de noter la présence des résistances R12 et R13 montées en "pull down" sur les lignes de donnée, en effet la norme USB impose l'utilisation de ces deux résistances pour un host.

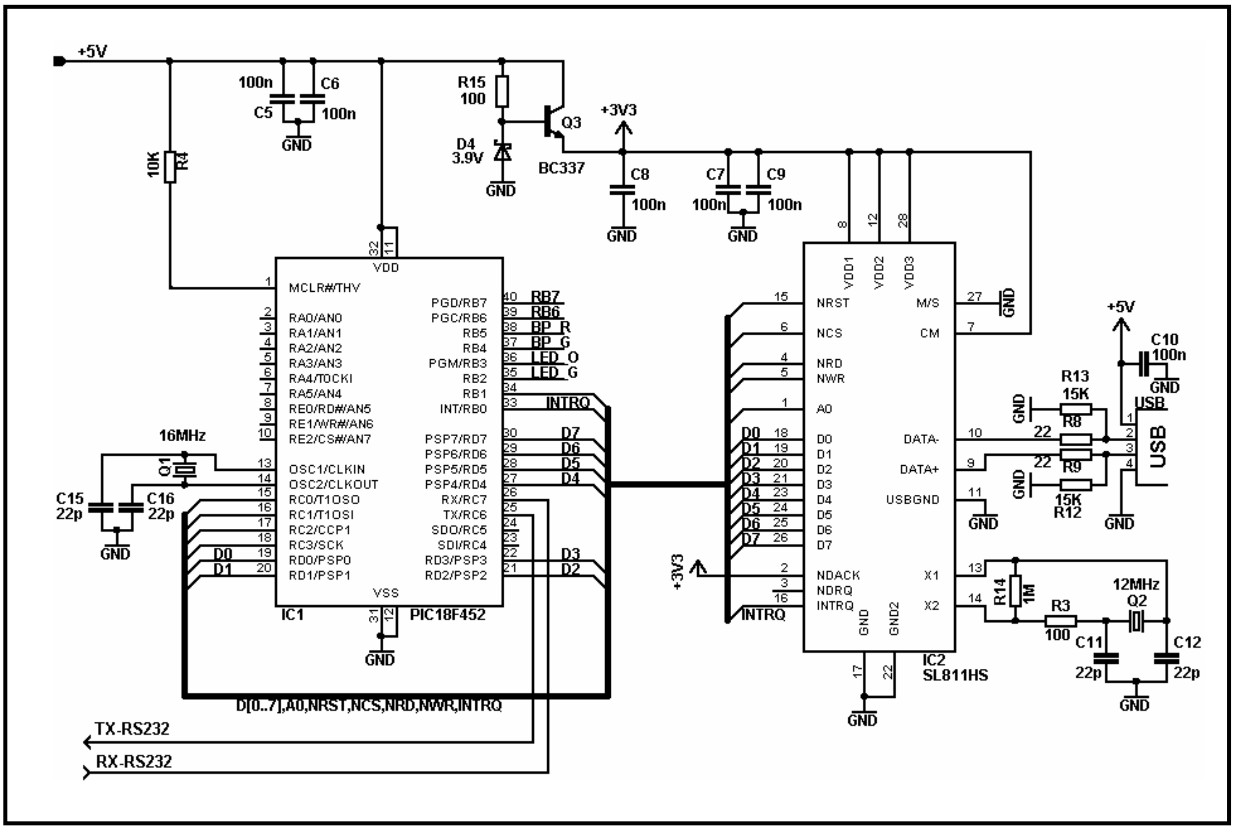

Figure 24 : Architecture de la carte

3.2.2 Implémentation des composants

❖ *Liste des composants*

Nom	Valeur	Nom	Valeur
R1	1K	C1	220u
R2	1K	C2	1u
R3	100	C3	100n
R4	10K	C4	100n
R5	470	C5	100n
R6	470	C6	100n
R7	470	C7	100n
R8	22	C8	100n
R9	22	C9	100n
R10	10K	C10	100n
R11	10K	C11	22p
R12	15K	C12	22p
R13	15K	C13	100n
R14	1M	C15	22p
R15	100	C16	22p
IC1	PIC18F452	D1	1N4004
IC2	SL811HS	D2	1N4148
IC3	7805	D3	1N4148
IC4	74AC14N	D4	zener 3.9V
Q1	quartz 16MHz	RS232	Sub D femelle 9 broches
Q2	quartz 12MHz	SUPPLY	Bornier 2 plots 5.08
Q3	BC337	USB	Connecteur USB type A

Figure 25 : liste des composants

❖ *Disposition des composants*

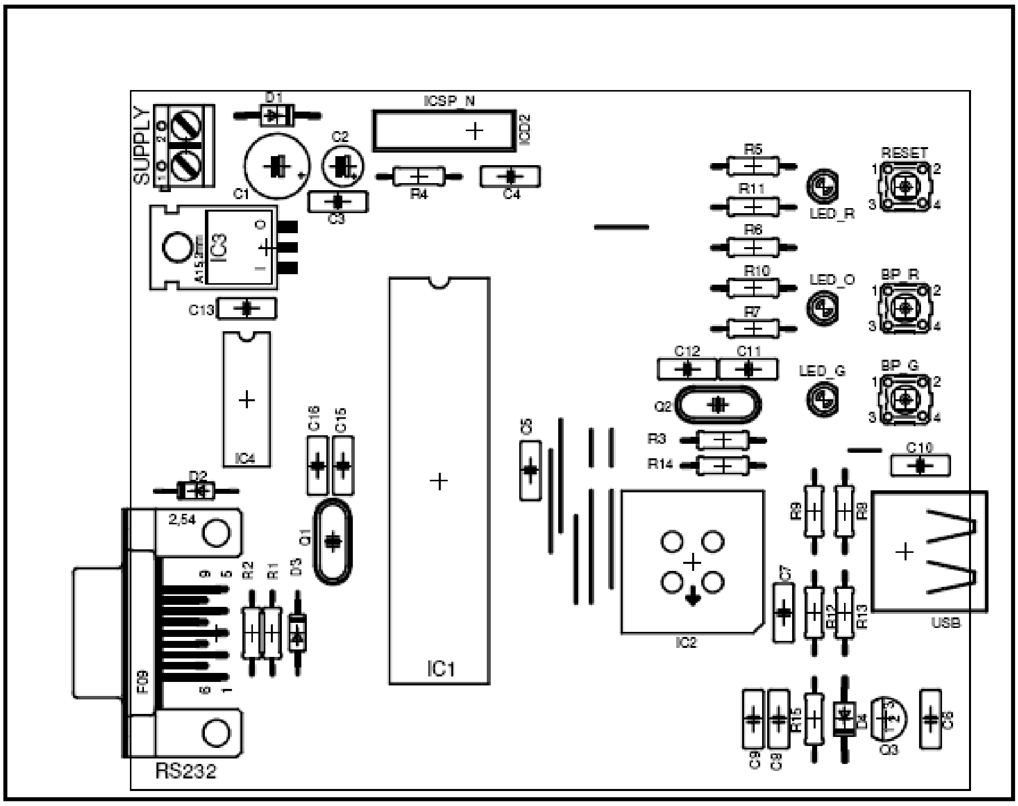

Figure 26 : Disposition des composants

❖ *Mise au point*

Il est important d'utiliser un support pour le PIC. Le premier test qui est à faire est de contrôler les tensions d'alimentation. Pour cela utilisez un bloc secteur 9 ou 12 V et connectez le sur le bornier "Supply". Contrôlez la tension 5V à la sortie du régulateur et le 3.3V sur

l'émetteur du transistor Q3.
Si ces tests sont concluants vous pouvez insérer le PIC et le SL811 dans leur support respectif.

3.3 Architecture de la carte

3.3.1 Fonctions d'interface avec le SL811HS

Il faut à présent se pencher sur la partie logicielle de mon host. Dans un premier temps il faut implémenter un certain nombre de fonctions qui permettront de communiquer avec le SEI. La communication avec le SEI est réalisée sur un bus de donnée 8 bits bidirectionnels. Cinq signaux sont nécessaires pour le contrôle des communications:

- CS : chip select pour activer le composant,
- WR : write pour effectuer une écriture,
- RD : read pour effectuer une lecture,
- RST : reset du SIE
- A0 : choix Commande/donnée

Vous pouvez remarquer que la pine d'interruption du SL811 est connectée sur RB0, cependant elle ne sera pas utilisée dans le programme. Pour réaliser un host cette pine ne me sera pas utile mais j'ai préféré la connecter.

Le SL811 possède de nombreux registres 8 bits. L'écriture/lecture se réalise en deux étapes principales : je commence par écrire sur le port de donnée l'adresse du registre à lire/écrire puis j'effectue une écriture/lecture du registre. Il faudra bien faire attention à positionner le port de donnée en sortie lors de l'écriture de l'adresse ou d'une donnée et en entrée lors d'une lecture d'une donnée.
Le SL811 permet aussi d'écrire/lire des registres en mode "burst", c'est à dire d'écrire/lire les octets les uns à la suite des autres sans avoir à spécifier l'adresse à chaque fois. Les fonctions sl811_write_buf(..) et sl811_read_buf(..) utilisent ce mode burst.

3.3.2 Fonctions d’hôte USB

Il s'agit maintenant de créer des fonctions qui réalisent l'énumération et le contrôle des périphériques MASS STORAGE. (Énumération). Comme mentionné au début, la première vérification à faire lors d'un branchement d'un périphérique est de détecter sa vitesse. Pour cela mon système doit détecter un positionnement de la ligne data+ ou data- au niveau haut. Mon host ne permet pas de contrôler les clés LOW SPEED pour des raisons de simplicité. Si je branche un périphérique LOW SPEED le programme sort de suite en échec. Si par contre le périphérique est un périphérique FULL SPEED (ou HIGH SPEED) comme attendu, après une temporisation (100ms) permettant une stabilisation de l'alimentation le microcontrôleur envoi une commande RESET_USB. Cette commande a pour effet de positionner les deux lignes data+ et data- au niveau bas pendant 20ms, le périphérique sait ainsi qu'il doit s'initialiser (reset usb définit dans la norme). La première requête USB à envoyer est comme vous avez pu le voir, une commande de changement d'adresse (SET_ADDRESS). En regardant cette fonction vous vous apercevrez que cette requête commence par envoyer un paquet SETUP avec les 8 octets correspondants à la requête SET_ADDRESS. Ensuite la fonction envoi un paquet IN auquel le périphérique répond par un paquet DATA vide pour dire que tout c'est bien passé.

La deuxième requête que je vous propose est la requête qui va permettre de vérifier qu'il s'agit bien d'un périphérique MASS_STORAGE. Cette requête c'est GET_CONFIGURATION. Remarquez que la commande GET_CONFIGURATION a été écrite pour récupérer plusieurs descripteurs à la fois, je récupère ici 32 octets. En fait, je récupère 4 descripteurs : le premier est le descripteur de CONFIGURATION, le second est le descripteur d'INTERFACE et les deux autres sont des descripteurs d'ENDPOINT. J'ai repéré les champs qui m’intéressent en les mettant en gras. Les octets que je récupère sont organisés comme vous le montre la figure 27.

OFFSET	Description
0	**bLength = 0x09**
1	**bDescriptorType = 0x02**
2 to 3	wTotalLength = 0x0020
4	bNumInterface
5	bConfigurationValue

6	iConfiguration
7	bmAttributes. Reserved
8	bMaxPower (x2 to have mA)
9	**bLength = 0x09**
10	**bDescriptorType = 0x04**
11	bInterfaceNumber
12	bAlternateSetting
13	bNumEndpoints
14	**bInterfaceClass = 0x08**
15	bInterfaceSubClass = 0x05
16	bInterfaceProtocol = 0x50
17	iInterface
18	**bLength = 0x09**
19	**bDescriptorType = 0x05**
20	**bEndpointAddress**
21	bmAttributes. TransferType
22 to 23	wMaxPacketSize
24	bInterval
25	**bLength = 0x09**
26	**bDescriptorType = 0x05**
27	**bEndpointAddress**
28	bmAttributes. TransferType
29 to 30	wMaxPacketSize
31	bInterval

Figure 27 : illustration des 32 octets de la commande GET_CONFIGURATION

Le host va contrôler la classe du périphérique, ici on vérifie qu'il s'agit de la classe MASS_STORAGE qui est repérée par la constante 0x08. Le host va ensuite lire le numéro des ENPOINTs qu'il doit utiliser pour envoyer et recevoir les données vers/depuis le périphérique de stockage de masse.

3.3.3 Présentation du code source

3.3.3.1Présentation

Le code que je fourni est fonctionnel pour effectuer une écriture ou une lecture sur une clé USB. Il se peut que certaines clés ne fonctionnent pas avec ce code : en effet malgré les normes USB et Mass Storage, il y a toujours des fabricants qui s'éloignent plus ou moins de ces normes

Le code fourni lit le secteur 0, il incrémente l'octet 508, et ré écrit le secteur modifié. Si j'ai choisi l'octet 508 du secteur c'est en fait pour ne pas altérer un octet du secteur 0 qui est un octet nécessaire au système de fichier (FAT ou autre). Évidement cette carte ne gère pas le système de fichier : La carte permet d'écrire ou de lire la mémoire flash secteur par secteur mais pas d'écrire un fichier ou un dossier.

L'objectif du projet étant de transférer tous les fichiers d'une clé A vers une clé B, il serait logique de faire le copiage à partir secteur 0. Avec cette méthode, le contenu de la clé B sera lisible sur un ordinateur.

3.3.3.2 Déclaration est constante

```
//=================================================================
=====================
//      Constant définitions
//=================================================================
=====================
#define     INTRQ       portb.f0
#define     NRST        portb.f1
#define     LED_G       portb.f2
#define     LED_R       portb.f3
#define     BP_R        portb.f4
#define     BP_N        portb.f5
#define     NWR         portc.f0
#define     NCS         portc.f1
#define     NRD         portc.f2
#define     A0          portc.f3
#define     SL811_DATA      PORTD

#define     READY                   0
```

```
#define 	COMPLETED 			0
#define 	NO_DEVICE 		1
#define 	NOT_FULL_SPEED 	2
#define 	NOT_MASS_STORAGE 	3
#define 	ERROR 			4
#define 	BUSY 			5
#define 	STALL 			6


//Macros
#define 	p_data_out(); {TRISD = 0x00;};
#define 	p_data_in(); {TRISD = 0xFF;};
```

3.3.3.3 Fonction main ()

```
void main (void)
{
	unsigned char tmp;
	unsigned long sector;

	// configuration des ports
		TRISA = 	0b11111111;
		TRISB = 	0b11110001;
		TRISC = 	0b11110000;
		TRISD = 	0b11111111;
		TRISE = 	0b00000111;

		ADCON1 	= 7; 		//activer digital I/O

	//initialisation des ports
		PORTA = 	0;
		PORTB = 	0;
		PORTC = 	0;
		PORTD = 	0;

		pause_ms(255);
		init232(8); 	//Initialiser UART 115200 8 avec un  oscillateur 16 MHz
		LED_G = 1;

		tx232_puts("\n\n\rStart program\n\r");

		sl811_init(); 		//SEI initialisation & attente du branchement du
périphérique

		LED_R = 1;

		tx232_puts("\n\rPlease BP_G to read a sector\n\r");
		while(BP_N == 1);
	//lire secteur
		sector = 0x00000000;
```

```
		sl811_read_sector(sector, buffer);

	//afficher secteur
		tx232_puts("\r\nSector 0x");
		tx232_long(sector);
		display_sector(buffer);

		tx232_puts("\n\rPlease BP_O to write modified sector\n\r");
		while(BP_R == 1);
	//Modifier secteur

		buffer[0x1FC] = buffer[0x1FC] + 1; //incrémentation : octet 508 du secteur

	//ecrire secteur

		sl811_write_sector(sector, buffer);

	//lire secteur
		sl811_read_sector(sector, buffer);

	//afficher secteur
		tx232_puts("\r\nSector 0x");
		tx232_long(sector);
		display_sector(buffer);

	while(1);		//fin du programme, réinitialiser le  PIC pour redémarrer

}
```

3.3.3.4 Mise en œuvre du hôte

Comme présenté dans la partie matérielle la carte host possède un port série RS232 qui va permettre de débuguer plus facilement. C'est pas obligatoire d'utiliser le port série pour faire fonctionner le montage, celui-ci servira simplement à afficher les données lues.

Pour débuguer, utilisez un logiciel terminal série (ex hyper terminal sous windows) et connectez votre carte sur le port série de l'ordinateur. Il faudra régler la configuration du port : 1 bit de start, 8 bits de données, 1 bits de stop, vitesse 115200 bits/s.

Conclusion

Tout au long, de ce chapitre j'ai présenté la démarche permettant d'aboutir à un système de transfert de fichier USB.

En effet, j'ai présenté le fonctionnement des différents contrôleurs implémentés sur ma carte, par la suite, j'ai exposé le code source qui permettrait de gérer le transfert de fichiers.

Il est vrai que la communication réelle et fonctionnelle entre 2 clés USB n'a pas été réalisée dans la pratique, cependant pour atteindre mon objectif initialement tracé, le recours à l'approche symétrique peut m'être d'une grande utilité dans le cadre de mes travaux de recherches ultérieures.

Conclusion générale

En ce $21^{\text{ième}}$ siècle, la technologie ne cesse de s'accroître et de se diversifier. Il est certain que cela est d'une grande importance et utilité dans la vie de l'être humain, mais cette diversité technologique n'était pas toujours accessible et pratique tel qu'attendu. Le problème se matérialisait dans le manque de compatibilité des différentes technologies.

La communication série n'a pas échappé à ce problème jusqu'à l'apparition de la norme USB qui a uniformisé cette communication.

Dans ce rapport j'ai exposé mon projet, consistant au transfert de fichier USB. Mais pour que mon projet soit retenu sur le marché, il faudra bien lui ajouter quelques aspects conviviaux tels qu'un écran LCD est des boutons de navigation pour sélectionner seulement les fichiers à copier. Cela, sous entend la présence d'un système de fichier tel que le FAT 16.

Un tel système s'avère d'une grande utilité dans tout domaine faisant appel au transfert de données USB en masse. Comme par exemple deux individus qui souhaitent s'échanger de la musique à partir de leurs lecteur MP3 ou MP4, pourront le faire sans avoir recours à un ordinateur; un autre exemple : quelqu'un qui veut transférer ses photos de son appareil vers sa clé USB, au lieu d'emporter avec lui tout un ordinateur, il n'aura qu'à utilisé le système de transfert de donnée puisque la majorité des supports de stockage de photo, tel que l'SD-card, sont soumis aux normes USB.

Ainsi je clôture ce rapport, dans l'espoir que mon travail puisse aider quelconque voulant se lancé dans la conception d'un système intégrant une interface USB Host/device.

Bibliographie

Progressez avec les microcontrôleurs pic de Gérard Samblancat.

Le bus USB de Xavier Fenard et Christophe Picaud.

USB Mass Storage : Designing and Programming Devices and Embedded Hosts de Jan Axelson

Webographie

http://www.microchip.com

Contenant tout les documents du constructeur des PIC.

http://www.cypress.com/

Contenant tout les documents du constructeur de l'SL811HS

http://claude.dreschel.free.fr/commentfaire/MPLABICD.htm

Pour avoir plus d'information sur le logiciel MPLAB.

http://micropic.free.fr/pic1.html

Pour avoir des informations sur la programmation des PIC.

http://g.fondeville.free.fr/usb.html

Pour avoir des information sur le protocole USB.

www.usb.org

Contenant toutes les description de la norme USB

Liste des figures

Annexe

/***/

```
//Write one byte in sl811 register
/******************************************************************************
*************/
void sl811_write(unsigned char adr, unsigned char value)
{
	p_data_out();
	NCS = 0;
	A0 = 0;
	SL811_DATA = adr;
	NWR = 0;
	asm("nop");
	NWR = 1;
	A0 = 1;
	SL811_DATA = value;
	NWR = 0;
	asm("nop");
	NWR = 1;
	NCS = 1;
	p_data_in();
}


/******************************************************************************
*************/
// Read one byte in sl811 register
/******************************************************************************
*************/
unsigned char sl811_read(unsigned char adr)
{
	unsigned char value;
	p_data_out();
	NCS = 0;
	A0 = 0;
	SL811_DATA = adr;
	NWR = 0;
```

```
	asm("nop");
	NWR = 1;
	A0 = 1;
	p_data_in();
	NRD = 0;
	asm("nop");
	value = SL811_DATA;
	NRD = 1;
	NCS = 1;
	p_data_in();
	return value;
}

/*************************************************************************
*************/
//Write buffer
/*************************************************************************
*************/
void sl811_write_buf(unsigned char adr, unsigned char * buffer, unsigned char size)
{
	unsigned char i;
	sl811_write(adr, buffer[0]);
	size--;
	i = 1;
	while(size != 0)
	{
		sl811_write_next(buffer[i]);
		i++;
		size--;
	}
}

/*************************************************************************
*************/
//Read buffer
```

```
/****************************************************************************
*************/
void sl811_read_buf(unsigned char adr, unsigned char * buffer, unsigned char size)
{
	unsigned char i;
	buffer[0] = sl811_read(adr);
	size --;
	i = 1;
	while(size != 0)
	{
		buffer[i] = sl811_read_next();
		i++;
		size--;
	}
}

/****************************************************************************
*************/
//Write without specify address
/****************************************************************************
*************/
void sl811_write_next(unsigned char value)
{
	A0 = 1;
	NCS = 0;
	p_data_out();
	SL811_DATA = value;
	NWR = 0;
	asm("nop");
	NWR = 1;
	NCS = 1;
	p_data_in();
}
```

```
/********************************************************************************
*************/
//Read without specify address
/********************************************************************************
*************/
unsigned char sl811_read_next(void)
{
	unsigned char value;
	NCS = 0;
	A0 = 1;
	p_data_in();
	NRD = 0;
	asm("nop");
	value = SL811_DATA;
	NRD = 1;
	NCS = 1;
	return value;
}
```

SOMMAIRE

Résumé

Le développement de la communication série a mit en valeur de nouveaux aspects d'échange de données. Le transfert de fichier, sujet de mon projet de fin d'études, consiste à transférer des fichiers d'un support de masse vers un autre via le port USB.

En première phase, j'étais conçu de réaliser un système embarqué permettant la communication entre une clé USB et un terminale de communication tel qu'un PC.

Pour ce fait, j'ai dû comprendre le fonctionnement du protocole USB et ainsi j'ai pu programmé un pic pour gérer cette communication.

J'ai dû aussi prévoir dans toute la réalisation de ce projet, la conception du système qui sera apte à la transmission des données entre deux clés USB sans intervention d'une machine performante dotée d'un système d'exploitation pourrant gérer cette transaction.

Mots clés : transfert, données, clé USB, système embarqué, PIC, communication.

Summary

The development of the communication series A emphasized new aspects of data exchange. The transfer of data, subject of my project of end of studies, consists in transferring from the files of a support of mass towards another via port USB.

In first phase, I had like stain the realization of an embarked system allowing the communication between a key USB and final of communication such as a PC.

For this fact, I had to include/understand the operation of protocol USB and thus I could programmed a PEAK to manage this communication.

The second phase, will consist in extending this project towards a transmission de<données between two keys USB without intervention of a powerful machine equipped with an operating system pourrant to manage this transaction.

Key words: transfer, data, key USB, embedded system, PIC, communication.

خلاصة

تنمية الاتصال مع سلسلة تسليط الضوء على جوانب جديدة من تبادل البيانات. نقل البيانات ، عن بلدي النهائية للمشروع ، هو لنقل الملفات من كتلة واحدة متوسطة الى آخر عن طريق ميناء USB.

في المرحلة الاولى ، وكما قلت اجراء بقعة على متن نظام يتيح الاتصال بين USB الرئيسية ومحطة الاتصالات مثل كمبيوتر شخصي.

ولهذا السبب ، كان لي لفهم اعمال USB من البروتوكول وذلك ولقد حدد موعدا لاجراء الموافقة المسبقه عن علم على ادارة هذا الاتصال.

المرحلة الثانية ستكون لتوسيع نطاق هذا المشروع لارسال <بيانات بين اثنين من مفاتيح USB دون تدخل من آلة قوية مع نظام التشغيل pourrant ادارة هذه العملية.

الكلمات الرئيسية : نقل البيانات ، USB نظام المجلس ، للموافقة المسبقه عن علم ، والاتصالات