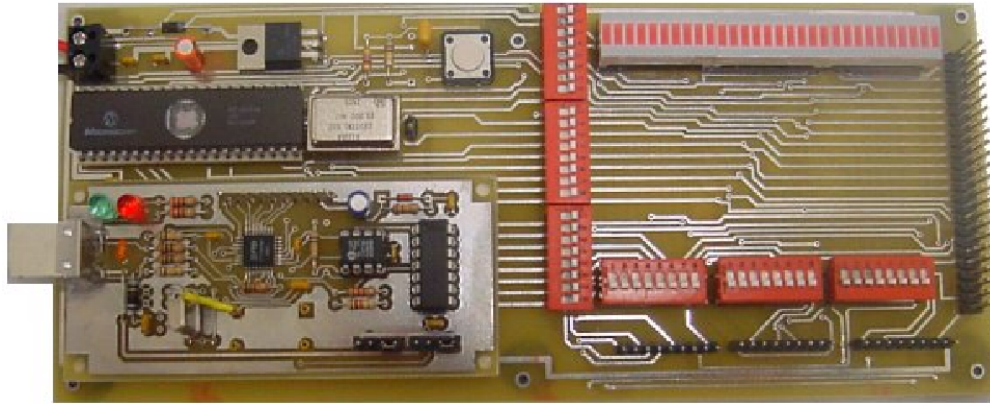


INTERFACE PCI4-USB

- Description du produit

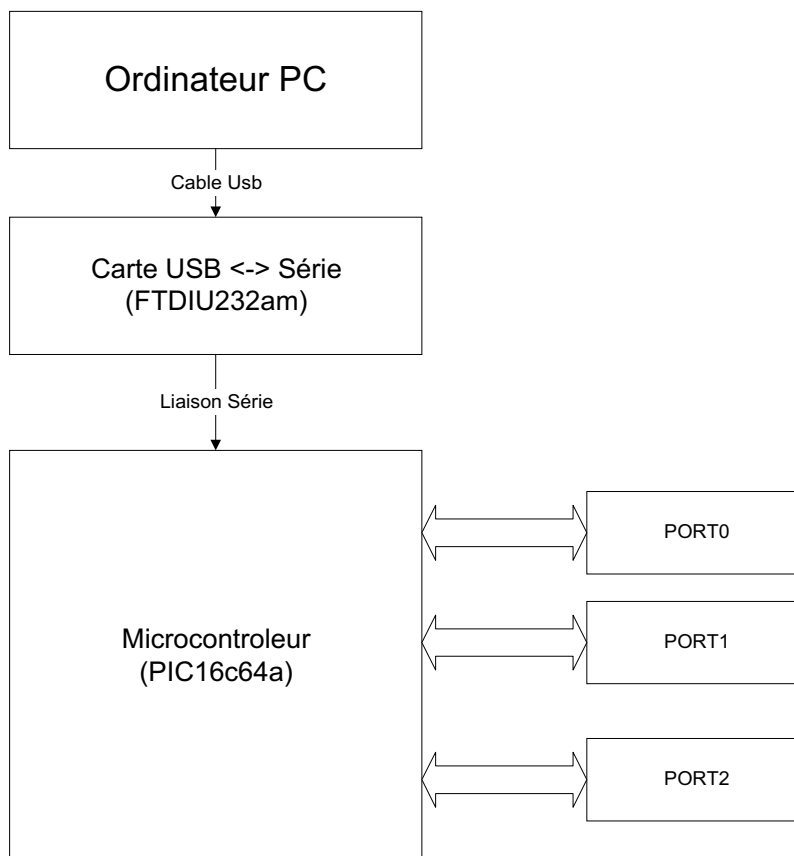


La carte PCI4 Usb est une carte contenant 24 entrées/sorties numériques. Cette carte possède aussi deux séries d'interrupteurs permettant de contrôler les sorties de 24 bits. Ces entrées peuvent être utilisées pour faire du contrôle numérique. La carte se relie par câble USB au PC. La carte utilise une DLL contenant un composant série qui permet d'envoyer les commandes sur le port série virtuel créé par le driver FTDI. À l'aide d'une combinaison de commandes et de données, nous pouvons écrire sur les ports de la carte ou lire un de ces ports. Les commandes correspondent à une action précise sur les ports du microcontrôleur. Le tableau ci-dessous vous montre les numéros de commandes ainsi de leurs utilités. Une DLL et un composant ActiveX fournit, permettrons à l'utilisateur d'utiliser les fonctions du protocole d'une façon simple et efficace. Les principales fonctions disponibles sont listées et expliquées dans la section DLL ou ActiveX de ce manuel.

Nom	Code 'HEX'	Fonction	Retour code 'HEX'
Détection	0x0A	Initialisation de la carte PCI4 Usb	0xAB
BitSet	0x2a	Positionne un bit à 1	-

Bit Clear	0x3a	Positionne un bit à 0	-
LireR0	0x4a	Lecture du Port0	Lecture R0
LireR1	0x4a	Lecture du Port1	Lecture R1
LireR2	0x4a	Lecture du Port2	Lecture R2
EcrireR0	0x5a	Écriture du Port0	-
EcrireR1	0x5b	Écriture du Port1	-
EcrireR2	0x5c	Écriture du Port2	-

Le schéma bloc de la carte PCI4 Usb est le suivant :



Premièrement, l'ordinateur PC servira à contrôler la carte PCI4-Usb. Pour ce faire, elle utilisera le Port Usb pour communiqué via le câble Usb au convertisseur Usb à Série. Le convertisseur Usb à Série convertira les données venant du Port Usb en

donnée série pouvant être lu par le microcontrôleur. Le microcontrôleur interprétera les données qu'il a reçues et exécutera les commandes s'il y a lieu. Les ports 0, 1 et 2 seront lu ou écrit par le microcontrôleur.

Procédure d'installation :

Matériel :

La procédure d'installation matérielle ne consiste qu'à relier l'interface Pci4-Usb vers l'ordinateur hôte. Le câble de liaison USB doit être configuré pour des connecteurs de Type A vers B.

Une clé dans les connecteurs USB permet d'interconnecter l'interface vers l'ordinateur sans inverser la polarité de la connexion. En somme, aucune erreur ne peut être faite en reliant l'interface Pci4-Usb vers le PC.

Logiciel :

L'installation logiciel se divise en deux parties : La partie gestionnaire de périphérique, 'Driver', pour la carte Pci4-USB⁽¹⁾ et la librairie 'DLL' ou 'ActiveX' qui permet l'accès à la carte par les logiciels de contrôles⁽²⁾.

1) Installation de l'Interface Pci4-USB

L'installation de l'Interface USB s'accomplit de façon automatique par Microsoft Windows 98/2000/Me. Après avoir relié l'interface USB sur un port de communication libre, le système d'exploitation effectue la lecture et la détection du

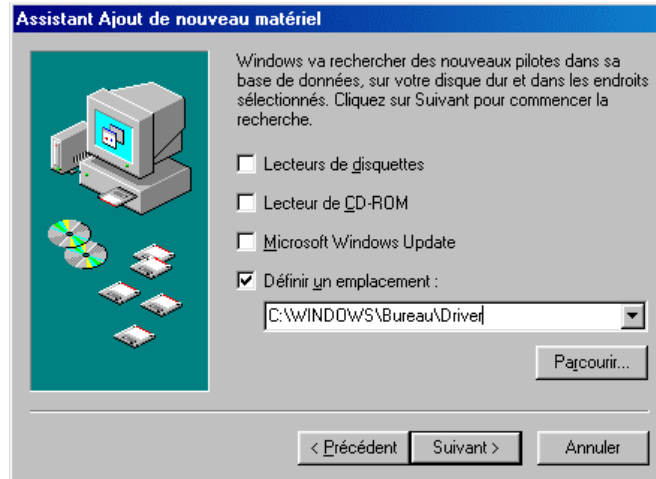
périphérique USB. Une boîte de dialogue, celle ci-dessous, vous informera sur la détection de l'interface USB au premier branchement.



Cette boîte de dialogue vous informera que l'interface Pci4-USB vient d'être détecté par Windows.



En cliquant sur suivant, il vous sera demandé de spécifier l'emplacement du gestionnaire de périphérique. La seconde option vous permettra de spécifier le 'Driver' selon un emplacement spécifique.



Les pilotes 'Driver' pour la carte Pci4-USB se retrouve dans le dossier E:\PCI4-USB\DRIVER\ ou 'E:\' représente la lettre de votre lecteur CD-ROM.

Le système d'exploitation vous confirmera par le message ci-dessous que les pilotes correspondant à la carte Pci4-USB sont conformes pour le type de périphérique détecté.



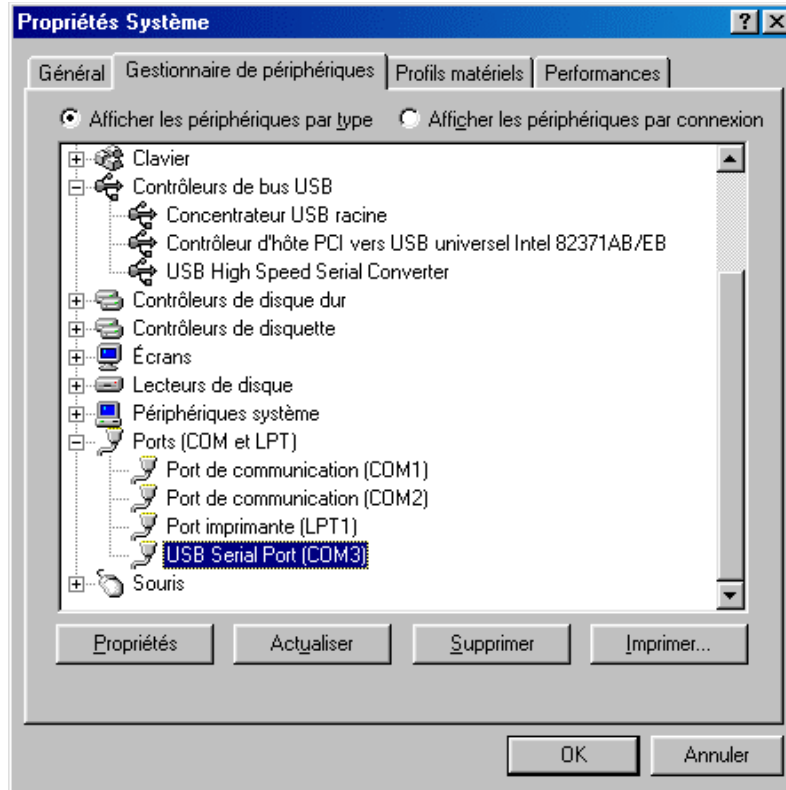
Une fois l'installation des pilotes effectués, un dernier message confirmera que l'interface 'USB High Speed Serial Converter' est maintenant présent sur le système.



La dernière étape consiste à effectuer l'installation du 'USB Serial Port' en suivant les mêmes étapes expliquées ci-dessus. Un dernier message vous confirmera de nouveau que le pilote est chargé et présent sur le système.



La vérification de l'installation peut se faire avec un clique-droit sur l'icône du bureau 'Poste de travail' et en sélectionnant propriétés.



Les propriétés systèmes, figure ci-dessous, vous informeront sur le périphérique détecté par le système d'exploitation. Les deux éléments 'USB High Speed Serial Convertor et USB Serial Port' confirme l'installation de l'Interface Pci4-USB sur le système.

2) Installation des bibliothèques 'DLL et ActiveX'

- DLL

La bibliothèque 'Dynamic Link Library' pour la carte Pci4-USB se retrouve dans le dossier E:\PCI4-USB\DLL\ ou 'E:\' représente la lettre de votre lecteur CD-ROM. La bibliothèque 'DLL' ne demande pas une installation particulière, donc elle doit être mise dans le répertoire du programme ou du projet en cours d'utilisation.

Pour désinstaller la librairie 'DLL', l'utilisateur doit supprimer le fichier *pci4.dll* du projet. Aucune modification de la base de registre ou création de fichier survient lors de l'installation de la librairie. En somme, la suppression de la librairie *pci4.dll* conclut la désinstallation.

- ActiveX

La librairie 'ActiveX' pour la carte Pci4-USB se retrouve dans le dossier E:\PCI4-USB\OCX\ ou 'E:\' représente la lettre de votre lecteur CD-ROM. L'installation automatique s'effectue en exécutant le fichier *installOcx.bat*. Ce fichier permet la copie du composant ActiveX vers le répertoire SYSTEM de Microsoft Windows, également il enregistre le composant auprès de Windows à l'aide de la commande *regsvr32.exe*.

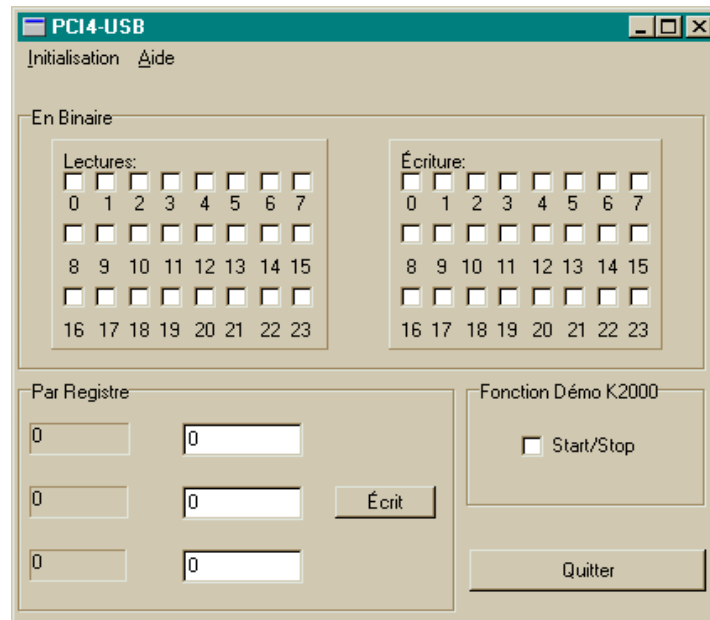
La seconde méthode consiste à copier manuellement le fichier *pci4usb.ocx* vers le répertoire SYSTEM de Windows. Par la suite, il faut exécuter la commande *regsvr32.exe pci4usb.ocx* pour compléter l'enregistrement du composant ActiveX.

En exécutant la commande *regsvr32.exe -u pci4usb.ocx* il est possible d'enlever l'enregistrement du composant ActiveX. Également, le composant doit être supprimé du répertoire SYSTEM de Windows : Fichier *pci4usb.ocx* et *pci4usb.oca*. La même opération peut être automatisée en exécutant le fichier *uninstallOcx.bat* présent dans le répertoire E:\PCI4-USB\OCX\ ou 'E:\' représente la lettre de votre lecteur CD-ROM.

Procédure d'opération

L'interface de test en C++ et en Visual Basic permet de tester les principales fonctions de la carte.

Interface de test en C++



Menu de l'interface :

Le menu Initialisation vous permet d'initialiser la carte. Normalement, l'Initialisation est effectuée lors du chargement de l'interface usager. Si la carte n'a pas été préalablement Initialiser, vous pouvez utiliser le sous-menu Initcarte.

Le menu aide vous permet de consulter la fenêtre A Propos qui vous renseignera sur les concepteurs de l'interface, du numéro de version, etc.

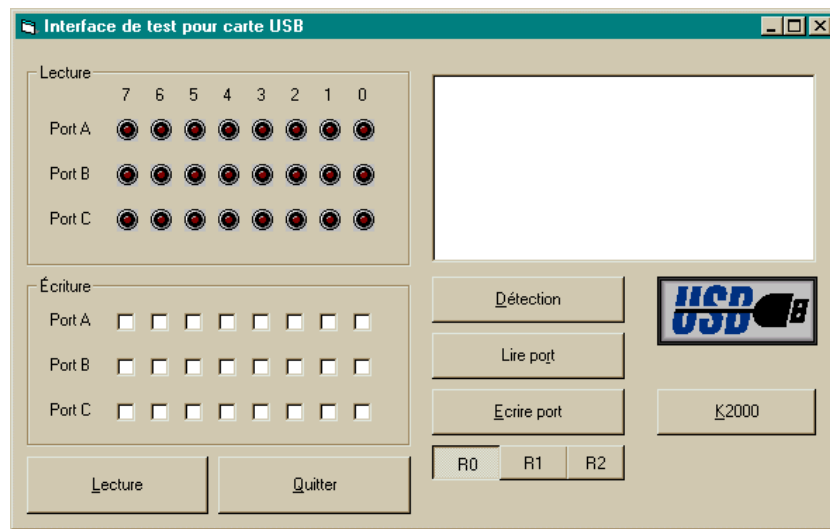
Deux sections vous sont offertes, la section En Binaire et la section Par Registre. La section En Binaire est répartie en deux groupes, le groupe de lectures et le groupe d'écritures. Le groupe de lecture permet de lire tous les ports bits par bits. Cette lecture se fait automatiquement grâce au temporisateur de lecture. La section Écriture permet de changer l'état de n'importe quel bit en cliquant sur la case à cocher représentant ce bit.

La section Par registrar vous permet de voir l'état de vos ports en décimal. Il vous permet aussi d'écrire une valeur directement sur un port. Pour ce faire, des boîtes

d'éditations vous permet d'écrire la valeur à écrire sur les ports 0, 1 et 2. Après avoir remplis les boîtes d'éditations, vous n'avez qu'à appuyer sur le bouton Écrit pour envoyer la valeur sur la carte PCI4-Usb.

La dernière section de l'application est la fonction de démonstration K2000. La fonction K2000 vous permet de faire une démonstration de la carte PCI4-Usb et de tester le bon fonctionnement des écritures sur la carte. La fonction de démonstration se démarre en cochant la case à cocher Start/Stop. La fonction arrêtera lorsque vous décochez la case à cocher Start/Stop.

Interface de test Visual Basic :



L'interface de test Visual Basic fonctionne de la même façon que l'interface de test en C++. Le bouton de détection vous permettra de détecter la carte. Un message apparaîtra dans la boîte de texte est vous indiquera si la carte a été détectée ou non. Le panneau Lecture vous indiquera l'état de chacun des bits des trois ports. Pour pouvoir visualiser l'état de chacun des bits, il faut appuyer sur le bouton Lecture. Le panneau Écriture vous permet d'affecter l'état de n'importe quel bit en cochant ou décochant la case correspondante au bit que vous voulez affecter. Le bouton Lire Port vous permet de lire un port spécifique. Vous pouvez choisir le port que vous voulez lire en utilisant les boutons R0, R1 et R2. Le résultat de cette lecture

sera afficher dans la boîte de texte dans le coin supérieur droit de l'application. De plus le bouton EcrirePort vous permet d'écrire la valeur 0xAA sur le port de votre choix que vous pouvez sélectionner en utilisant les boutons R0, R1 ou R2. Une confirmation de cette écriture apparaîtra dans la boîte de texte.

Exemple d'utilisation :

Si vous désirez tester votre carte PCI4 Usb, voici quelques étapes d'utilisation de l'interface de test en Visual Basic et en C++.

Premièrement, vous devez détecter votre carte.

En C++ : La détection se fera automatiquement lorsque vous lancerez l'application. Un message vous indiquera si la détection a réussi. Si la détection a échoué, utiliser le sous-menu InitCarte du menu d'Initialisation. Si après quelques essais, la carte n'est pas détectée, référez-vous à la section d'installation de ce manuel.

En Visual Basic : Vous devez appuyer sur le bouton Détection pour lancer la détection de la carte. Un message apparaîtra dans la zone de texte et vous indiquera si la détection a réussi ou non. Si la carte n'a pas été détectée, réappuyez sur le bouton Détection. Référez-vous à la section d'installation de ce manuel en cas de problème de détection. Lorsque votre carte sera détectée, vous pouvez utiliser les possibilités de l'interface de test.

Exemple d'écriture sur un bit :

Pour écrire sur un bit, il suffit de cocher ou décocher la case du bit que vous voulez écrire. En C++ et en Visual Basic, le panneau Écriture vous permet d'effectuer cette action.

Exemple : En C++ et en Visual Basic, coché la case du bit 8 du panneau écriture.

Exemple de lecture d'un bit :

Pour l'état d'un bit : En C++ : La lecture se fera automatiquement et vous devriez voir l'état du bit 8 modifié précédemment dans le panneau Lecture.

En Visual Basic : Appuyer sur le bouton Lecture pour mettre à jours l'affichage de l'application. Vous devriez voir changer l'état du bit 8 modifié précédemment.

Pour écrire une valeur sur un port :

En C++ : Dans la section Par port, entrer votre valeur dans la boîte d'édition correspondant au port désiré. Appuyer sur le bouton écrit. Il est important de mentionner que les trois boîtes d'éditions sont envoyées vers les ports. Si vous ne désirez pas qu'un port soit affecté, mettez la valeur présente de ce port dans la boîte d'édition avant d'appuyer sur le bouton Ecrit.

En Visual Basic : Sélectionner le port voulu en utilisant les boutons R0, R1 et R2. Appuyer sur le bouton Ecrire Port. La valeur 0xAA sera écrite sur le port choisi.

Pour lire une valeur sur un port :

En C++ : La lecture se fait automatiquement. Vous n'avez qu'à vérifier les boîtes à cocher du panneau lecture ou les boîtes du groupe par port. Ils vous indiqueront les valeurs de chacun des ports.

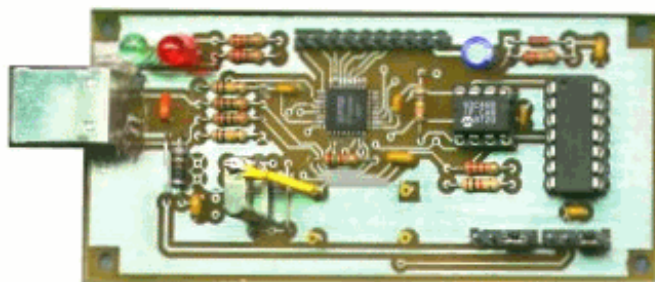
En Visual Basic : Sélectionner le port voulu en utilisant les boutons R0, R1 et R2. Appuyer sur le bouton Lire Port. La valeur du port qui a été lu sera affichée dans la boîte d'édition.

Contenu matériel de l'interface Pci4-USB

L'interface Pci4-USB se divise en deux circuits. Le premier circuit étant le convertisseur USB <-> Série et le second, la carte d'entrée/sortie avec le microcontrôleur PIC. Une description pour chaque circuit est disponible ci-dessous. Également, le dernier paragraphe de cette section explique l'interrelation entre les deux cartes : Carte convertisseur USB et Microcontrôleur PIC avec 24 entrées/sorties.

- Convertisseur USB

La première carte, celle illustré ci-dessous permet la conversion du protocole USB vers le protocole série RS232.

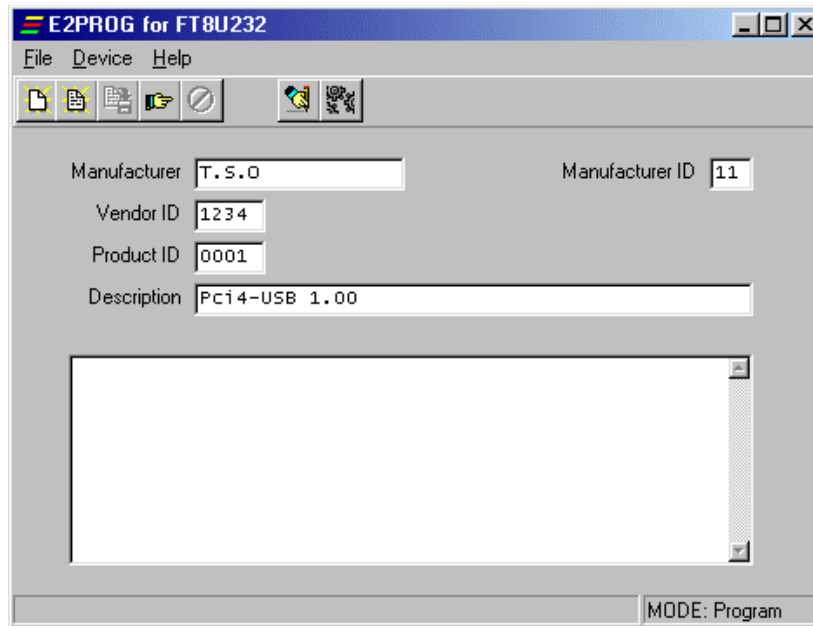


Le cœur de cette carte consiste en un circuit intégré provenant de la compagnie FTDI. Ce circuit intégré, le FT8U232AM, initialise la communication avec l'ordinateur hôte en envoyant les descripteurs nécessaires pour configurer la connexion de l'Interface USB vers le PC. Son rôle est d'effectuer la détection de la carte auprès du système d'exploitation et de contrôler les trames et protocole USB

pour les convertir en un format standard au RS232. Un schéma électrique en annexe ou sur le CD-ROM 'E:\SCHEMA\USB' son disponible pour illustrer en détail le fonctionnement du circuit

La première section du convertisseur USB <-> Série est la liaison des lignes de communications différentielles 'D+/D-'. Chaque ligne ont une impédance créé par les résistances R12 et R14. La valeur de ces résistances est spécifiée par les standard USB disponible à l'adresse www.usb.org ou sur le document *usbStd.pdf* présent dans le répertoire E:\DOC\ . Également, la résistance R4 de type PULL-DOWN présente sur la ligne 'D-' fixe la communication USB à un débit de 1.2 MB/s pour le mode LOW-SPEED.

Une seconde partie très importante du convertisseur USB <-> Série est la mémoire série de type Microwire connecté sur le circuit spécialisé. Son but est de personnaliser le descripteur du périphérique USB. Il est possible de programmer directement cette Eeprom en circuit par l'utilitaire E2Prog . L'utilitaire se retrouve sur le site du fabricant www.ftdi.co.uk et dans le répertoire E:\UTILS\EEPROM\ . Cependant, le projet existant n'utilise pas cette fonction pour personnaliser le descripteur de l'Interface Pci4-USB. L'interface graphique ci-dessous provient de l'utilitaire en question.



En somme, un descripteur par défaut est chargé si le descripteur présent en Eeprom est invalide ou si l'Eeprom est manquant du circuit. Donc, le circuit est présentement configuré pour utiliser le descripteur par défaut puisque la mémoire Eeprom ne contient présentement pas de descripteur spécifique ou personnalisé.

Le connecteur présent sur la carte permet d'inter relier le convertisseur USB <-> Série vers l'interface des entrées/sorties. Le connecteur se compose principalement de 10 broches reliées directement vers les entrées/sorties du circuit FT8U232AM. La fonction de chaque broche est décrite ci-dessous dans le tableau. Toutes les broches sont utilisées par des signaux de type RS232 '+5v/0v' et l'alimentation du circuit.

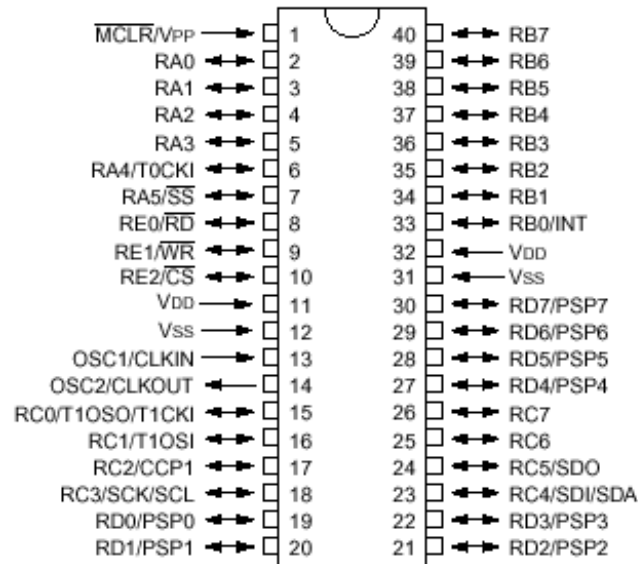
Position	Nom
1	TXD
2	RXD
3	RTS
4	CTS
5	DTR

6	DSR
7	DCD
8	SLEEP
9	Vcc
10	Gnd

Tous les signaux disponibles sur le connecteur au format RS232 ne sont pas différentiels. Donc, leur tension est de type logique $5v = 1$ et $0v = 0$. Également des diodes électroluminescentes 'LED' permettent la visualisation des changements sur les lignes TX et RX.

- Interface microcontrôleur

Le microcontrôleur utilisé pour interfacer les 24 entrées/sorties et la carte de conversion USB provient de la compagnie Microchip. Le microcontrôleur PIC16c64a, illustrer ci-dessous possède l'architecture Harvard et peut contenir jusqu'à 2K de code en Eprom. Le modèle utilisé dans le projet de l'interface USB est effaçable par une lampe à l'ultra violet, par ajout des 'One Time Programable' devront être utilisé pour la production puisqu'il ne sont pas dispendieux.



PIC16C64A

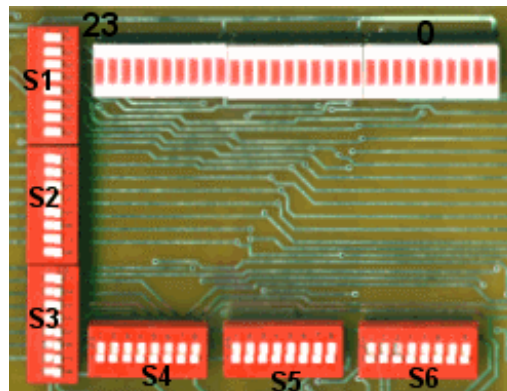
La tâche du microcontrôleur PIC est d'interpréter les commandes reçus par les lignes de communication série selon un protocole expliqué dans la partie logiciel. Après avoir traduit les commandes et les données transmises par le convertisseur USB, il modifiera ou lira l'état des 3 ports 8 bits pour contrôler les 24 entrées/sorties. Le tableau ci-dessous affiche l'assignation des ports 0-2 sur le microcontrôleur PIC.

Nom	Connexion sur PIC
Port 0	RB0-RB7
Port 1	RC0-RC7
Port 2	RD0-RD7

Les 24 entrées/sorties sont capables de délivrer une intensité 'Ampère' suffisante pour alimenter directement une del 'LED' ou une entrée de type CMOS/TTL. Cependant, une suralimentation ou une décharge statique sur l'une des 24 lignes pourrait détruire le microcontrôleur. C'est pourquoi il est conseillé d'utiliser un

étage de type 'buffer' ou 'opto coupleur' pour protéger le microcontrôleur des accidents pouvant subvenir sur les lignes.

Une partie du circuit permet l'observation de l'état de chaque entrées/sorties avec l'aide de del 'LED'. Les 24 del 'LED' présentes sur le circuit, image ci-dessous, affichent l'état des lignes. Également, il est possible de changer l'état des entrées en déplaçant la position des interrupteurs S4-S6. Les interrupteurs S1-S3 sont utilisés pour l'activation ou la désactivation d'un port ou d'un bit en particulier.

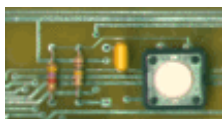


L'alimentation de la carte Pci4-USB, image ci-dessous, est fournie principalement par un bloc d'alimentation qui est raccordé sur le connecteur P7. La tension doit obligatoirement se situer en 6v et 9v avant d'entrer dans le régulateur '7805' qui



fixera la tension à 5v. Une diode D1 évite les inversions sur l'alimentation et protège le régulateur. Finalement, un filtrage est effectué par les condensateurs C1-3 avant et après le régulateur.

Le circuit de reset, image ci-dessous, force l'entrée MCLR 'pin 1' à 0 pour



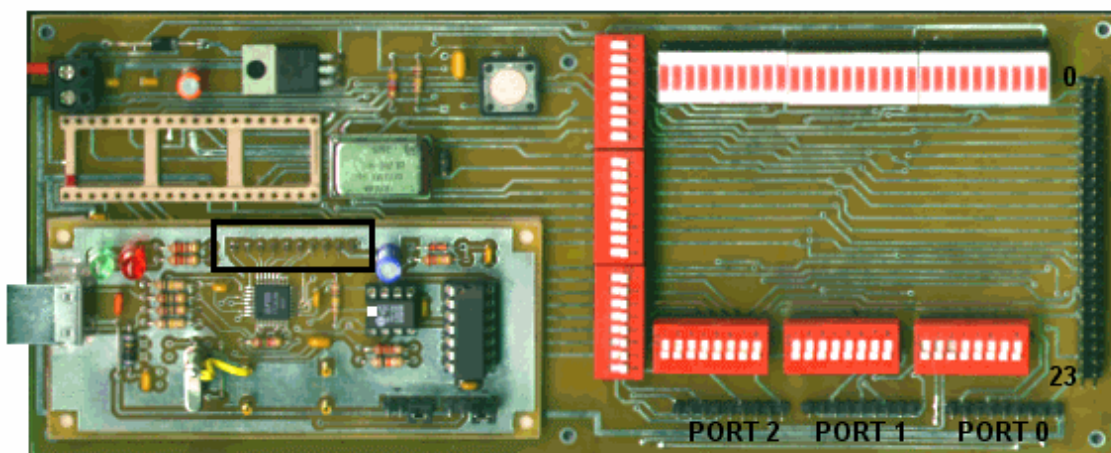
réinitialiser le microcontrôleur. Également, le micro interrupteur S7 force l'entrée MCLR à 0 et le circuit RC assure un délais suffisant pour effectuer un reset complet.

Pour conclure sur la section de l'interface microcontrôleur, le fonctionnement reste brièvement simple puisque le microcontrôleur gère le fonctionnement de toutes les entrées/sorties présentes sur la carte. Seul le circuit de visualisation, d'alimentation et de reset sont ajoutés pour compléter le fonctionnement de la carte.

*Pour de plus amples renseignements sur le microcontrôleur, la fiche technique du PIC16c64a 'pic16c64.pdf' est disponible sur le CD-ROM dans le répertoire E:\DOC\PIC.

Interrelation entre les cartes

Le circuit imprimé du convertisseur USB et de l'interface avec microcontrôleur PIC s'assemble via le connecteur série '10 broches', celui encadré par le rectangle, pour compléter l'interface Pci4-USB. Le résultat de l'assemblage est illustré ci-dessous.



Pour faciliter l'accès aux entrées/sorties, des borniers sont accessibles sur le côté du circuit imprimé. Chaque borniers sont relié au Port 0-2. Également, une broche du connecteur permet d'utiliser la masse du circuit.

Le tableau ci-dessous représente le brochage des connecteurs disponibles sur le côté.

Position	Nom
1	Bit 0
2	Bit 1
3	Bit 2
4	Bit 3
5	Bit 4
6	Bit 5
7	Bit 6
8	Bit 7
9	Gnd 'Masse'

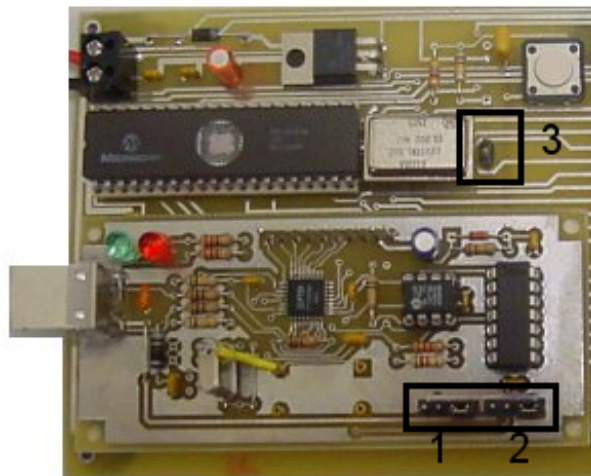
Les trois ports de 8 Bits sont disponibles sous un seul connecteur. Le connecteur de 48 broches situé au bout du circuit permet l'utilisation d'un câble nappe pour interconnecter l'interface pci4-USB sur un projet. Tous les bits présent sur le connecteur possède une masse pour interfacer les bits séparément et permettre une réduction des parasites sur la communication.

Le tableau ci-dessous identifie tous les bits présent sur le connecteur 48 broches situés au bout de la carte d'interface.

Position	Nom
1	Bit 0
2	Gnd
---	---
47	Bit 23
48	Gnd

- Ajustement interne de la carte

La carte pci4-USB possède en tous 3 cavaliers configurables par le technicien. Ces trois cavaliers servent principalement à configurer le type d'alimentation et la vitesse du cristal. L'image ci-dessous illustre les trois cavaliers présent sur la carte.



Les cavaliers 1 et 2 ont la possibilité d'être mis selon deux positions: [--]-- ou --[--] ou le symbole '-' représente la broche et '['] le cavalier. Cependant, le troisième connecteur ne possède pas de position. Il peut être retiré ou posé pour spécifier le type d'alimentation. Pour clarifier les configurations possibles, le tableau ci-dessous explique les fonctions disponibles.

Cavalier 1	
Options	Fonction
Gauche [--]--	Cristal de 48 MHz utilisé 'Multiplicateur interne inactif'
Droite --[--]	Cristal de 6 MHz utilisé 'Multiplicateur interne actif'

Cavalier 2	
Options	Fonction
Gauche [--]--	Alimentation du convertisseur série fournit par le câble USB
Droite --[--]	Alimentation du convertisseur série fournit par la carte microcontrôleur

Cavalier 3	
Options	Fonction
Fermé [--]	Alimentation du convertisseur série fournit par la carte microcontrôleur
Ouvert --	Alimentation du convertisseur série fournit par le câble USB 'Ne nécessite pas l'alimentation de la carte'

En somme, le cavalier 2 et 3 sont utilisés conjointement pour spécifier le type d'alimentation. Si l'utilisateur décide d'alimenter sa carte USB à l'aide du bloc d'alimentation et non par le câble USB, le cavalier 2 doit être à droite et le cavalier 3 est fermé pour alimenter le convertisseur USB.

Avec l'aide des cavaliers il serait possible d'auto alimenter la carte pci4-USB par le port USB, mais une limitation de 500 mA s'impose. Donc, il est fortement

conseillé d'utiliser une alimentation externe provenant d'un bloc d'alimentation. L'intensité 'Ampère' fournit par le câble USB est insuffisant pour le fonctionnement de la carte USB dû aux nombreuses DEL 'Led' utilisées pour visualiser les changements sur les bits.

- Banc de test

Les tests effectués en laboratoire sont surtout sur la carte convertisseur série et la carte microcontrôleur PIC. La première carte, carte USB, peut être testée simplement en utilisant un cavalier qui sera connecté sur la broche 1 et 2 du bornier de 10 broches présent sur la carte. En inter reliant ces deux broches un NULL-MODEM est créé à partir de l'interface USB. Donc, il est possible d'utiliser un programme de communication 'ex : Hyperterminal' ou le programme terminal présent sur le CD-ROM dans le répertoire E:\UTILS\TERMINAL\ pour communiquer directement avec la carte USB. Un test fonctionnel devrait retourner correctement tous les caractères envoyés par l'utilisateur.

En cas d'erreur le fichier debug.pdf présent dans le dossier E:\DOC\FTDI\ peut être utilisé pour dépanner le circuit du convertisseur USB. Également, des modifications sur le schéma électrique sont effectuées en tout temps par la compagnie FTDI. Donc, il est conseillé de visiter le site Internet www.ftdi.co.uk pour obtenir les dernières informations et mise à jour du circuit.

Les tests matériels sur l'interface microcontrôleur PIC sont restreints. La première étape consiste à vérifier si l'alimentation est présente sur les broches du microcontrôleur PIC. La seconde est d'effectuer un test à l'aide d'un câble série RS232 'COM1 ou COM2' et de gérer le protocole de communication manuellement à l'aide d'un logiciel de terminal. Cependant, pour effectuer un test de communication RS232 un convertisseur de tensions $\pm 12v$ est exigé. En laboratoire, un circuit spécialisé le MAX233 semblable au MAX232 a été utilisé

pour réaliser l'interface de test. En somme, les tests présentés ci-dessus permettent d'isoler les problèmes pouvant subvenir.

Contenu logiciel du projet

Fonction de la DLL

OnTimer:

Cette méthode est appelée par l'événement OnTimer du temporisateur. Cette fonction permet de sortir le PC de sa boucle d'attente lorsque que la carte ne répond pas.

TestDevice:

Cette méthode détecte la présence de la carte et sur quel COM virtuel il faut lui envoyer les données. Retournera l'état logique vrai si la carte est détectée ou l'état logique faux si elle n'est pas détectée.

OnRxChar:

Cette méthode est appelée par l'événement OnRxChar lorsque le composant série reçoit un caractère. On indiquera alors qu'une donnée a été reçue.

OnTxEmpty:

Cette méthode est appelée par l'événement OnTxEmpty lorsque le tampon de transmission est vide. Cette méthode sert à la validation des commandes et des données par la ligne de contrôle DTR. Des problèmes ont été remarqués lors de

l'implémentation, car le driver FTDI du convertisseur USB à SERIE ne semble pas géré correctement cet événement. TxEmpty

TxDonnee(char cCar):

Cette méthode est utilisée pour envoyer une donnée vers la carte PCI4 Usb. Cette méthode a été utilisée en remplacement de la validation des commandes et données via la ligne de contrôle DTR. Le but de cette méthode est de s'assurer que l'on a bien transmis une commande avant de transmettre la donnée.

TxCommande(char cCar):

Cette méthode est utilisée pour envoyer une commande vers la carte PCI4 Usb. Cette méthode a été utilisée en remplacement de la validation des commandes et données via la ligne de contrôle DTR. Le but de cette méthode est de s'assurer que l'on peut transmettre la commande afin d'éviter qu'elle soit prise pour une donnée.

bit_set(int bit):

Permet de mettre le bit spécifié sur la carte PCI4 Usb à un.

bit_clear(int bit):

Permet de mettre le bit spécifié sur la carte PCI4 Usb à zéro.

int lire_bit(int bit):

Permet de lire un bit spécifié sur la carte PCI4 Usb. On retournera l'état (0 ou 1) du bit.

`ecrire_bit(int bit, int etat):`

Permet de mettre le bit spécifié selon l'état spécifié. Les états possibles sont 0 ou 1.
Cette méthode utilisera les méthodes `bit_set` et `bit_clear`.

`unsigned int lire_R0:`

Permet de lire le Port0, soit les bits 0 à 7 de la carte PCI4 Usb.

`unsigned int lire_R1(void):`

Permet de lire le Port1, soit les bits 8 à 15 de la carte PCI4 Usb.

`unsigned int lire_R2:`

Permet de lire le Port2, soit les bits 16 à 23 de la carte PCI4 Usb.

`ecrire_R0(int nPort):`

Permet d'écrire sur le Port0, soit sur les bits 0 à 7 de la carte PCI4 Usb.

`ecrire_R1(int nPort):`

Permet d'écrire sur le Port1, soit sur les bits 8 à 15 de la carte PCI4 Usb.

`ecrire_R2(int nPort):`

Permet d'écrire sur le Port2, soit sur les bits 16 à 23 de la carte PCI4 Usb.

Fonction de l'interface C++

FormActivate:

Lors de l'ouverture de l'application de test, la fonction FormActivate est appelée. Cette fonction aura pour but de détecter la carte PCI4 Usb et de démarrer le temporisateur de lecture de la carte.

Timer1Timer:

Cette méthode sera appelée par le temporisateur de lecture sur un OnTimer. Dans cette méthode, on lira les trois ports de la carte et on affichera les valeurs dans les cases prévues à cet effet et dans les cases à cocher de la section lecture.

Chkwb2Click:

Cette méthode sera appelée lorsque que l'on click dans une case à cocher du groupe d'écriture. On écrira ensuite la valeur sur la carte PCI4 Usb en utilisant la propriété bit.

btnEcritClick:

Cette méthode sera appelée par le bouton Ecrit. Il permettra d'écrire sur la carte PCI4 Usb les valeurs contenues dans les boîtes d'édition.

mnuInitClick:

Cette méthode est appelée par le menu InitCarte de l'application et permet de détecter la carte. Cette méthode appellera la fonction FromActivate.

mnuAProposClick:

Cette méthode permet de voir la fenêtre A Propos qui donne quelques informations sur l'application de test.

ChkStartClick :

Cette méthode permet de démarrer la fonction K2000. Cette méthode est une méthode créer pour la démonstration du fonctionnement de la carte.

tmrK2000Timer:

Cette méthode est appelée par le tmrK2000 lors de l'événement OnTimer. Cette méthode décale un bit de gauche à droite ou de droite à gauche. Cette méthode est une méthode créée pour la démonstration du fonctionnement de la carte.

btnQuitterClick:

Cette méthode vous permet de quitter l'application.

- Description des événements interceptés

OnTimer:

a) Programme de test:

1- L'événement OnTimer du temporisateur tmrTimer utilisé pour effectuer une lecture a un temps donnée.

2- L'événement OnTimer du temporisateur tmrK2000 utilisé pour un décalage de bit à un intervalle précis.

b) DLL:

L'événement du tmrTimer est utilisé dans deux situations:

1- Lors de la détection l'événement OnTimer permettra d'indiquer que le temps de réponses est terminé.

2- Lors d'une lecture, l'événement OnTimer sortira le PC de sa boucle d'attente une trop longue attente.

OnClick

a) Chkwb2Click

Événement OnClick des cases à cocher. Cet événement servira à écrire l'état d'un bit dont le numéro est contenu dans le tag de la case à cocher.

b) btnEcritClick

Événement OnClick du bouton qui permet d'écrire sur la carte PCI4 Usb les valeurs contenues dans les boîtes d'éditations.

c) mnuInitClick

Événement OnClick du sous-menu Init. Permet d'initialiser la carte PCI4 Usb

d) mnuAProposClick.

Événement OnClick du sous-menu A Propos. Permet de voir la fenêtre A Propos.

e) chkStartClick

Événement OnClick de démarrer ou d'arrêter la fonction K2000 .

f) btnQuitterClick

Événement OnClick permettant de fermer l'application de test.

OnRxChar:

L'événement OnRxChar du composant série de la DLL permet de recevoir des caractères du port série.

OnTxEmpty

L'événement OnTxEmpty nous indique que le tampon de transmission du composant série est vide.

- Donnée membre stratégique

- DLL

char cRegR0 : Contendra la dernière valeur lue du port 0.

char cRegR1: Contendra la dernière valeur lue du port 1.

char cRegR2: Contendra la dernière valeur lue du port 2.

int m_nCommande: Indiquera si l'on doit transmettre une donnée ou une commande à zéro, elle indiquera que l'on peut transmettre une commande à un elle indiquera que l'on peut transmettre une donnée

int m_nFlag: Indiquera s'il y a eu détection de la carte.
À zéro, elle indiquera que la carte n'a pas été détectée.

À un, elle indiquera que la carte a été détectée.

int m_nDataReady: Indiquera qu'une donnée a été reçue à zéro, elle indiquera qu'aucune donnée n'a été reçue à un, elle indiquera qu'une donnée est disponible.

int m_nTimeOut: Indique que le délai permis pour une réception est dépassé
À zéro, elle indiquera que le temps d'attente n'est pas dépassé
à un, elle indiquera que le temps d'attente est dépassé.

- Programme de test

int m_nBit : Numéro du bit où l'on est rendu pour la fonction K2000

int m_nDir: Direction, droite ou gauche pour la fonction K2000

À zéro, elle indiquera que l'on va à droite.

À un, elle indiquera que l'on va à gauche.

- Index des fonctions utilisés

Nom de la fonction	Fichier contenant le code	Fichier contenant les prototypes
FormActivate	fpci4.cpp	fpci4.h
Timer1Timer	fpci4.cpp	fpci4.h
Chkwbit2Click	fpci4.cpp	fpci4.h
btnEcritClick	fpci4.cpp	fpci4.h :
mnuInitClick	fpci4.cpp	fpci4.h
mnuAProposClick	fpci4.cpp	fpci4.h
chkStartClick	fpci4.cpp	fpci4.h
tmrK2000Timer	fpci4.cpp	fpci4.h
btnQuitterClick	fpci4.cpp	fpci4.h

OnTimer	pci4dll.cpp	pci4dll.h
TestDevice:	pci4dll.cpp	pci4dll.h
OnRxChar:	pci4dll.cpp	pci4dll.h
OnTxEmpty:	pci4dll.cpp	pci4dll.h
TxDonnee(char cCar):	pci4dll.cpp	pci4dll.h
TxCommande(char cCar):	pci4dll.cpp	pci4dll.h
bit_set(int bit):	pci4dll.cpp	pci4dll.h
bit_clear(int bit):	pci4dll.cpp	pci4dll.h
int lire_bit(int bit):	pci4dll.cpp	pci4dll.h
ecrire_bit(int bit, int etat):	pci4dll.cpp	pci4dll.h
unsigned int lire_R1(void):	pci4dll.cpp	pci4dll.h
unsigned int lire_R2:	pci4dll.cpp	pci4dll.h
ecrire_R0(int nPort):	pci4dll.cpp	pci4dll.h
ecrire_R1(int nPort):	pci4dll.cpp	pci4dll.h
ecrire_R2(int nPort):	pci4dll.cpp	pci4dll.h

- Diagramme de classe



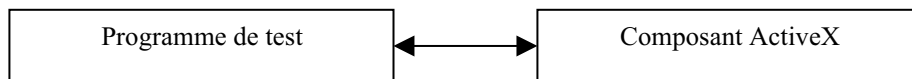
CPCI3 : Classe principale de la DLL Pci3dll gérant la carte PCI4-Usb.

Fonction du composant ActiveX

- Composant ActiveX

Le logiciel Microsoft Visual Basic a été utilisé pour la création du composant ActiveX pci4usb. Ce composant est utilisable dans les logiciels de développement tel que Microsoft Visual Basic, Borland C++ Builder et Delphi. Le code source de ce composant est disponible sur le CD-ROM dans le répertoire E:\CODE\OCX\ ou E:\ représente la lettre de votre lecteur CD-ROM.

Le fonctionnement de chaque fonction est décrit par les ordigrammes situés en annexe et ci-dessous. Également, le diagramme de classe ci-dessous illustre l'implémentation du composant ActiveX dans un projet. Le projet utilisé pour contrôler l'interface Pci4-USB consiste en un programme de test qui effectue des opérations sur les entrées/sorties.



- Description des fonctions et méthodes

Lire_Port(*nPort* As Integer) As Byte

Cette fonction effectue la lecture du Port 0-2 en envoyant 0x4a selon le protocole établi pour la communication. Le paramètre *nPort* doit contenir le numéro du port, soit : 0,1,2. La fonction transmettra la fonction ainsi que la commande avant de recevoir le résultat. Le résultat de la lecture sera retourné dans la variable *Lire_Port* sous un byte.

Ecrire_Port(*nPort* As Integer, *nData* As Integer)

La fonction *Ecrire_Port* permet l'écriture d'un byte sur le Port 0-2. Le paramètre *nPort* doit contenir le numéro du port, soit : 0,1,2 et le paramètre *nData* contient la valeur qui sera écrite directement sur le port du microcontrôleur. Cette fonction

utilise la commande 0x5a pour transmettre une donnée. Le numéro du port est ajouté au numéro de la fonction pour spécifier le port choisi, par exemple : Port 0 = 0x5a, Port 1 = 0x5b, Port 2 = 0x5c.

Bit_Set(nBit As Integer)

Le positionnement d'un bit est effectué en utilisant la fonction *Bit_Set*. Le numéro du bit est spécifié par le paramètre *nBit* qui doit être compris entre 0-23. Donc, pour activer le bit 0 du Port 1 il faut mettre *nBit* à 8. La fonction 0x2a est utilisé pour spécifier le nom de la fonction au microcontrôleur.

Bit_Clear(nBit As Integer)

Le positionnement d'un bit est effectué en utilisant la fonction *Bit_Clear*. Le numéro du bit est spécifié par le paramètre *nBit* qui doit être compris entre 0-23. Donc, pour désactiver le bit 0 du Port 2 il faut mettre *nBit* à 16. La fonction 0x3a est utilisé pour spécifier le nom de la fonction au microcontrôleur.

Init_Usb() As Boolean

Une détection de l'Interface Pci4-USB est réalisée en utilisant la fonction *Init_Usb*. Cette fonction retournera **true** si l'interface Pci4-USB est connectée sur le port USB de l'ordinateur. La fonction effectue une recherche de l'interface en envoyant la commande 0x0a. Par la suite, une lecture est effectuée pour vérifier si la capture donne 0xab. Le code 0xab retourné par le microcontrôleur spécifie si l'interface est bien connectée et présente sur l'ordinateur. En somme, cette fonction permet de déterminer si une interface de communication pci4-USB est présentement connecté et prêt à communiquer. Le résultat sera faux si l'interface est inactive.

Stop_Usb()


Cette fonction permet la fermeture du port de communication avec l'interface pci4-USB. En appelant cette procédure, la communication sera coupé instantanément. Également, cette procédure est appelé à la destruction du composant ActiveX. L'utilisateur n'est pas obligé de faire appel à cette procédure, car son appel est fait durant fermeture du composant ActiveX.

- Index des fonctions

Fonction	Fichier source
Public Function Lire_Port(nPort As Integer) As Byte	Objet ActiveX usrUsb 'usrUsb.ctl'
Public Function Ecrire_Port(nPort As Integer, nData As Integer)	Objet ActiveX usrUsb 'usrUsb.ctl'
Public Function Bit_Set(nBit As Integer)	Objet ActiveX usrUsb 'usrUsb.ctl'
Public Function Bit_Clear(nBit As Integer)	Objet ActiveX usrUsb 'usrUsb.ctl'
Public Function Init_Usb(nComm As Integer) As Boolean	Objet ActiveX usrUsb 'usrUsb.ctl'
Public Function Stop_Usb()	Objet ActiveX usrUsb 'usrUsb.ctl'
Public Sub APropos()	Objet ActiveX usrUsb 'usrUsb.ctl'

- Description des événements

Sub usbTest_Click()

Le composant ActiveX ne possède pas d'événement spécifique pour la communication avec l'interface pci4-USB. Cependant, l'événement *Click* peut être utilisé pour appeler une référence d'aide créée par le programmeur si l'utilisateur effectue un clic sur l'image, celle-ci , du composant ActiveX.

- Description des données membres

bInterfaceOk As Boolean

La variable booléenne *bInterfaceOk* permet de connaître, sans repasser par la fonction *Init_Usb*, si l'interface pci4-USB a été détectée correctement. La valeur **true** signifie que l'interface est présente et active sur l'ordinateur. Par contre, l'interface est inactive si *bInterfaceok* est égale à **false**.

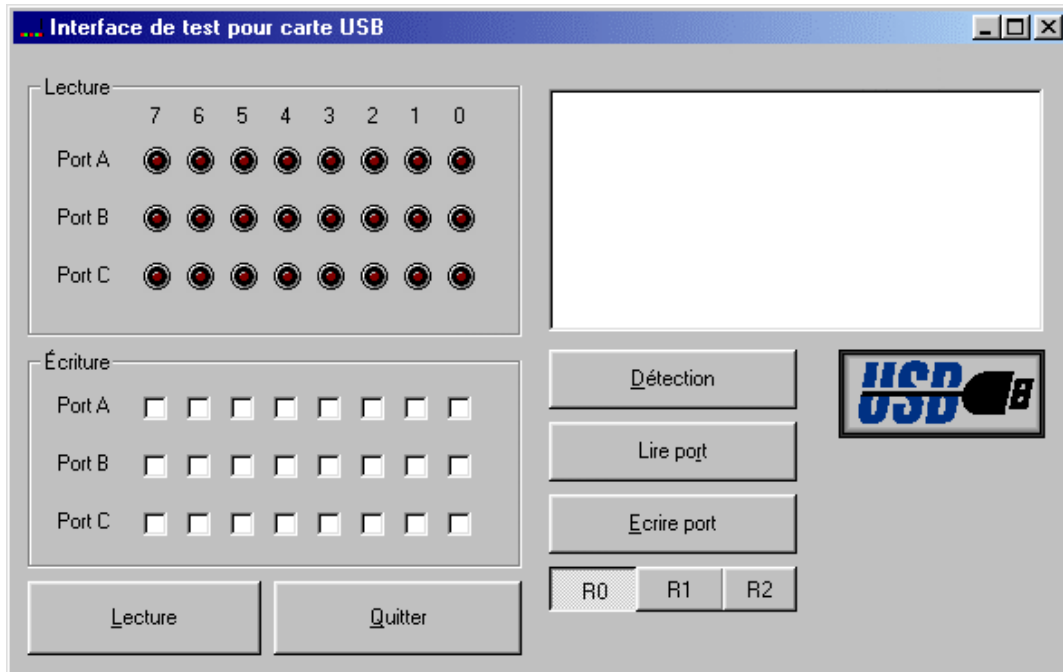
bTimeOut As Boolean

La communication avec l'interface pci4-USB utilise des délais dans les communications en entrées pour éviter toutes erreurs qui pourrait causer une boucle sans fin. La variable *bTimerOut* peut être utilisé pour créer une fonction personnalisée avertissent l'utilisateur que l'interface pci4-USB vient d'être déconnecté du port USB. La variable booléenne *bTimeOut* est égale à **true** si le temps d'attente pour une réponse dépasse les limites permises. Cependant, un résultat **false** signifie que l'interface est toujours présente.

- Programme de test

Un programme de test, utilisant le composant ActiveX pci4usb.ocx, est répertorié dans le dossier E:\TEST\Vb\ ou E:\ représente la lettre de votre lecteur CD-ROM.

Pour utiliser ce programme il suffit d'exécuter le programme *prjTestInterface.exe* après avoir installé correctement le composant ActiveX auprès de Microsoft Windows. Également, il est possible d'utiliser directement les sources du programme de test en ouvrant le groupe de projet *grpOcx.vbg* sous Microsoft Visual Basic 6. L'interface graphique est illustrée ci-dessous.



L'interface graphique permet la lecture, l'écriture sur les ports ou des bits en utilisant le composant ActiveX *pci4usb.ocx* pour accéder l'interface et le microcontrôleur. Avant d'effectuer une communication, l'utilisateur doit cliquer sur *Détection* pour procéder à l'Initialisation de la carte USB. Une fois initialiser correctement, l'utilisateur peut travailler avec les fonctions de communications.

- Fonction du PIC16C64A

InitPic:

Fonction qui permet de remettre tous les ports du pic à 0 et choisir la direction (Entrée ou Sortie) des ports.

Bit_Set(char cBit):

Fonction permettant de mettre le bit spécifié à un.

Bit_Clr(char cBit):

Fonction permettant de mettre le bit spécifié à zéro.

Ecri_Port(char cPort, char cData):

Fonction permettant d'écrire la valeur spécifiée sur le port spécifié.

unsigned char Lire_Port(char cPort):

Fonction permettant de lire un port spécifié. On retournera la valeur lue sur ce port.

Setup:

Permet de faire l'initialisation de la communication.

TxNextBit:

Permet de transmettre les prochains bit à être transmit.

PutChar(char):

Permet d'envoyer une valeur sur le port série.

GetChar:

Permet de recevoir un caractère venant du port série.

StartBitDetect:

Permet la détection du StartBit lors d'une réception série.

RcvNextBit:

Permet de recevoir le prochain bit qui doit être reçu.

Nom de la fonction	Fichier contenant le code	Fichier contenant les prototypes
InitPic:	usbSerie.c	usbSerie.c
Bit_Set(char cBit):	usbSerie.c	usbSerie.c
Bit_Clr(char cBit):	usbSerie.c	usbSerie.c
Ecri_Port	usbSerie.c	usbSerie.c
Lire_Port	usbSerie.c	usbSerie.c
Setup:	usbSerie.c	usbSerie.c
TxNextBit:	usbSerie.c	usbSerie.c
PutChar(char):	usbSerie.c	usbSerie.c
GetChar:	usbSerie.c	usbSerie.c
StartBitDetect:	usbSerie.c	usbSerie.c
RcvNextBit:	usbSerie.c	usbSerie.c

- Donnée membre stratégique

char TxByte: Contiendra le caractère a transmettre.
char RxByte; Contiendra le caractère reçu.
char TxRxBitCount: Contiendra le nombre de bit reçu ou transmit.
char SerialStatus: Variable qui servira de registre de statut pour le port série
char cControlBit: Variable qui contiendra l'état de la ligne de contrôle qui
 servira à valider si on reçoit une donnée ou une commande.

- Programme de test

Le programme de test s'exécute en utilisant le programme pci4.exe. Ce programme de test a été créé pour être capable de voir l'état des ports du pic et aussi de pouvoir écrire des valeurs sur la carte d'interface. Aussi, dans le but de tester toutes les caractéristiques de la carte, nous avons créer deux zones, une zone de lecture et une zone d'écriture. La zone de lecture vous permet d'afficher les valeurs soit bit par bit en utilisant la zone de case à cocher consacré à la lecture. Aussi, des boîtes vous indiqueront les valeurs lues par port. Une zone d'écriture est aussi disponible et vous permettra d'écrire un bit à la fois ou d'écrire sur tous les ports. La zone de case à cocher vous permet en cochant ou décochant une case à cocher d'affecter un bit à la fois. Aussi des boîtes d'éditions permettront d'écrire directement sur les ports de la carte en appuyant sur le bouton Ecrit. Une fonction K2000 vous permettra de faire une démonstration de la carte. Cette fonction créera un déplacement vers la droite ou vers la gauche sur l'affichage à LED de la carte.

- Programme d'ajustement du baudrate

Deux programmes d'ajustement du baudrate sont disponibles. Ces programmes visent à ajuster les constantes BaudRate et BaudRatePlusHalf.

Pour utiliser le programme trouvebaud.c

1- Programmer le PIC16c64 avec le programme `trouvebaud.c`. Pour savoir la procédure a utilisé pour la compilation du programme et la programmation du pic, se référé à la section Procédure de développement.

2- Utilisé une application terminal série qui permettre un ajustement rapide des paramètres de communication.

3- Ouvrir le port série virtuel créer par le pilote FTDI. Pour connaître le port série virtuel, il suffit d'ouvrir l'item système du panneau de configuration et dans la section Port (COM et LPT) repéré le com portant comme nom USB Serial Port . Configurer votre terminal avec les paramètres 8 bit de donnée, 1 stop bit, pas de parité et baudrate auquel vous voulez configurer le pic.

4- Mettre la carte d'interface sous-tension après avoir replacé le PIC16c64 à sa place. Si la carte est déjà sous-tension, effectuer une remise à zéro.

5- Plusieurs valeurs seront envoyer vers le PC. Ces valeurs auront le format suivant Xyyy. La position X contiendra la valeur de test « (0xab ou 171). La position y contiendra le caractère qui contiendra la valeur de la constante BaudRate a ajuster, répété trois fois. Trouver la trame parmi les trames envoyées. Utilisé une table ASCII pour découvrir la valeur du baudrate en décimal. Si vous ne trouvez pas la valeur de test, il se peut que l'horloge de la carte d'interface ne supporte pas le baudrate choisie.

6- Remplacer la valeur de la constante dans le programme principale de l'interface.

Pour utilisé le programme `trouvebaudplus.c`

- 1- remplacer la valeur de la constante BaudRate dans le programme trouvebaudplus.c avec celle trouver précédemment dans le programme trouvebaud.c.
- 2- Programmer le PIC16c64 avec le programme trouvebaudplus.c. Pour savoir la procédure a utilisé pour la compilation du programme et la programmation du pic, se référé à la section Procédure de développement.
- 3-Utilisé une application terminal série qui permettre un ajustement rapide des paramètres de communication.
- 4- Ouvrir le port série virtuel créer par le pilote FTDI. Pour connaître le port série virtuel, il suffit d'ouvrir l'item système du panneau de configuration et dans la section Port (COM et LPT) repéré le com portant comme nom USB Serial Port . Configurer votre terminal avec les paramètres 8 bit de donnée, 1 stop bit, pas de parité et baudrate auquel vous voulez configurer le pic.
- 5- Mettre la carte d'interface sous-tension après avoir replacé le PIC16c64 à sa place. Si la carte est déjà sous-tension, effectuer une remise à zéro.
- 6- Transmettre n'importe quel caractère. Utiliser toujours le même caractère. Après plusieurs essaie vous devriez recevoir une trame ayant le format xYYY où x est le caractère émis et Y sera la valeur de la constante BaudRatePlusHalf répété à trois reprise.
- 7- Remplacer la valeur de la constante dans le programme principale de l'interface.

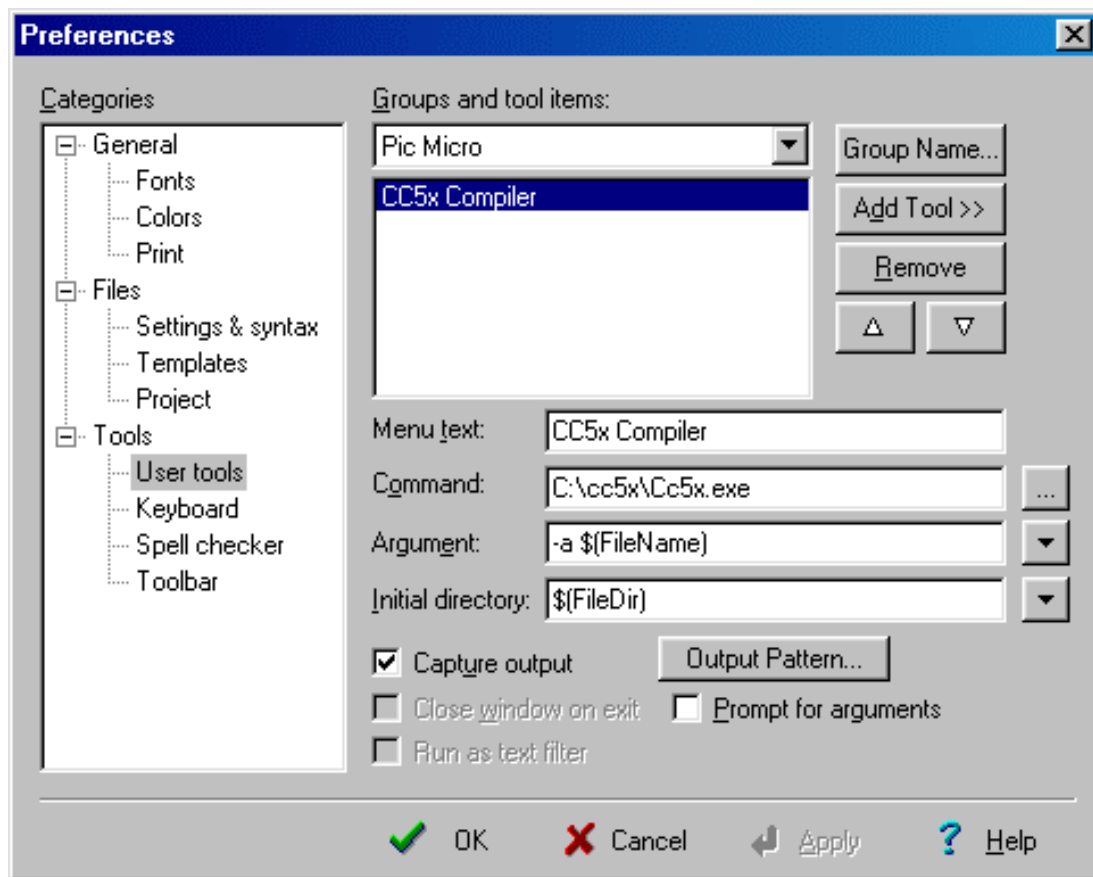
Après avoir trouver la valeur des constantes choisie, l'interface devrait être fonctionnel au BaudRate choisie.

Procédure de développement

Pour le développement logiciel un compilateur c vers assembleur provenant de la compagnie B Knudsen Data a été utilisé. Le compilateur CC5x, présent sur le CD-ROM dans le répertoire E:\UTILS\CC5X\, permet la compilation de code jusqu'à un maximum de 1024 octets pour la version éducative.

Pour installer le programme CC5x, il suffit de décompresser le fichier *cc5x.zip*, présent dans E:\UTILS\CC5X\ ou E:\ représente la lettre du CD-ROM, sur le disque dur. La seconde étape consiste à configurer un programme d'édition.

La compilation d'un fichier source s'effectue sous une boîte MS-DOS, mais le programme EditPlus a été utilisé pour faciliter l'édition et la compilation du code. L'image ci-dessous illustre la configuration utilisé avec EditPlus pour compiler un fichier source avec CC5x



Cependant, un autre programme d'édition tel que 'UltraEdit' pourrait être utilisé. Il faut alors spécifier l'argument '-a' au compilateur CC5x pour assembler un fichier source.

Le fichier *cc5x.pdf* au format 'Acrobat' est disponible sur le CD-ROM dans le répertoire E:\DOC\CC5X\ . Il explique en profondeur les options de compilations et les instructions en C pour le microcontrôleur PIC.

- Recompilation du programme

La recompilation du programme, code C vers assembleur, se fait en utilisant le compilateur CC5x. Une fois l'installation du compilateur fait avec l'éditeur

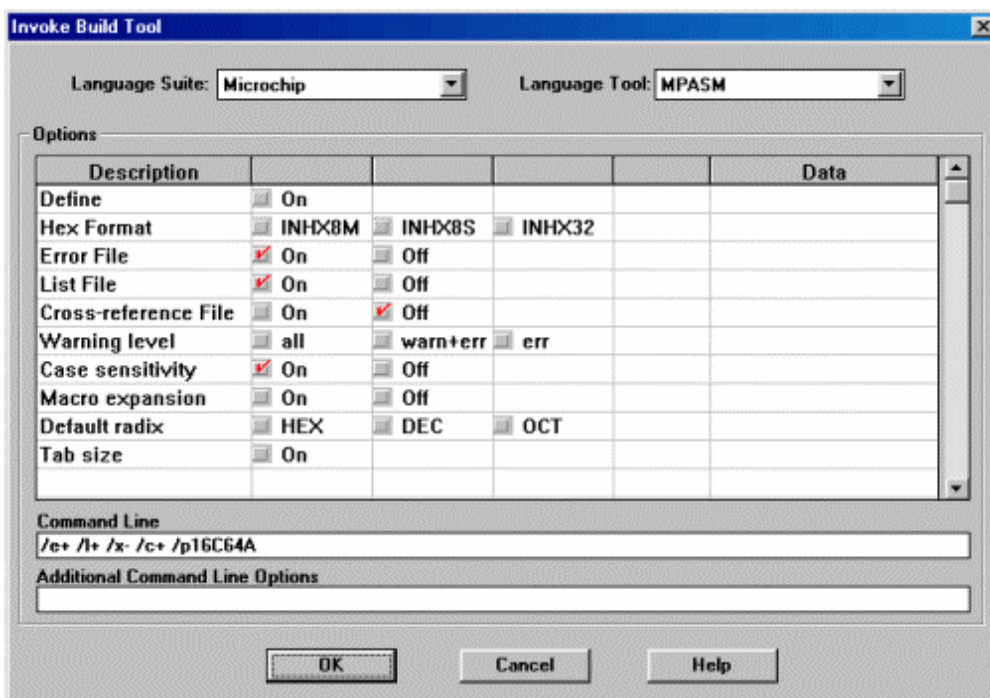
‘Expliqué dans l’étape de développement’ , le code C peut être recompilé par le menu Tools->CC5x make. Le résultat en assembleur sera mis dans le dossier du fichier source. La compilation du code assembleur vers l’opcode est expliqué dans l’étape de la Mise en ROM.

- Mise en ROM

Pour la mise en ROM, soit en mémoire morte, le logiciel MpLab doit être utilisé pour transmettre le programme vers le programmeur PicStart plus.

La procédure ci-dessous peut être utilisé pour programmer le microcontrôleur PIC16c64 servant dans l’interface USB. La première étape consiste à compiler le code en assembleur ‘*.asm’ pour obtenir les opcode ‘*.hex’ nécessaire pour la programmation de l’Eprom à l’intérieur du microcontrôleur.

Le chargement du fichier assembleur se fait par le menu File->Open et non par l’icône de la barre d’outils. Une fois le fichier ouvert, il doit être compilé en code machine ‘*.hex’ par le menu Project->Build Node. La boîte de dialogue ci-dessous apparaîtra pour configurer la compilation.



En cliquant sur OK, le fichier se compile en code machine selon le type de microcontrôleur choisi.

La seconde étape consiste à connecter le programmeur PicStart Plus sur un port série disponibles, COM1 ou COM2. Une fois branché, il faut l'activer via l'option du menu PICSTART Plus -> Enable programmer. Après avoir reconnu le programmeur PicStart, le logiciel ouvrira une fenêtre pour configurer le type de microcontrôleur, le type d'oscillateur et quelques options supplémentaire pour le fonctionnement interne du microcontrôleur. Les options choisies pour l'Interface USB sont :

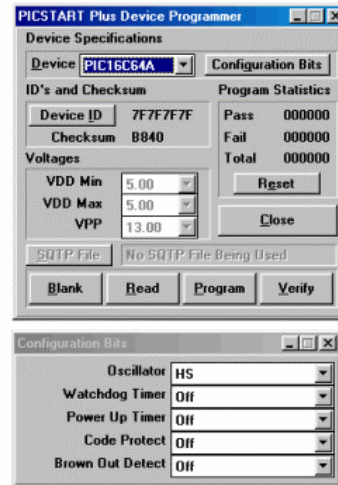
Option	Valeur	
Oscillator	<u>HS</u>	Oscillateur HS, XT pour crystal
Watchdog Timer	Off	Inactif
Power Up Timer	Off	Inactif
Code Protect	Off	Actif pour production
Brown Out Detect	Off	Inactif

Après s’être assuré que le microcontrôleur est bien effacé, en choisissant l’option ‘Blank’, le microcontrôleur peut être reprogrammé avec le nouveau code. Pour démarrer la programmation, il faut cliquer le bouton ‘Program’. Une boîte de dialogue affichera l’avancement de la programmation.

```

0010 2825 082F 00A3 3002 05A3 082F 00A4 3010  %/.../..
0018 05A4 08A3 1903 281E 2107 2825 1283 08A4  ....%..
0020 1003 2824 2200 2825 2218 1283 0E21 0083  .$.%..$.
0028 0EA0 0E20 0009 2001 2194 1283 14AF 21E3  ..)....
0030 1283 00A6 082F 00A8 3008 05A8 1003 282F  ..)..../
0038 0826 3A2A 1903 28AF 3A10 1903 285B 3A70  &*..0..[p
0040 1903 2867 3A10 1903 287E 3A01 1903 288B  ..g..~...
0048 3A07 1903 2899 3A56 1903 28A7 282F 21E3  ..U.../..
0050 1283 00A6 082F 00A8 3008 05A8 1003 282F  ..)..../
0058 0826 20CA 282F 21E3 1283 00A6 082F 00A8  &./.../..
0060 3008 05A8 1003 282F 0826 2110 282F 21E3  ..)/R.../
0068 1283 00A6 082F 00A8 3008 05A8 1003 282F  ..)..../
0070 0826 216E 1283 00A6 0826 21A1 1283 082F  &n...&../
0078 00A7 3001 05A7 1903 2876 282F 21E3 1283  ...U.../..
0080 00A6 082F 00A8 3008 05A8 1003 282F 01A9  ./.../..
0088 0826 2153 282F 21E3 1283 00A6 082F 00A8  &S/.../..
0090 3008 05A8 1003 282F 3001 00A9 0826 2153  /...&S/..
0098 282F 21E3 1283 00A6 082F 00A8 3008 05A8  /.../...
00A0 1003 282F 3002 00A9 0826 2153 282F 30A8  ./...&S/..
00A8 21A1 1283 082F 00A7 3001 05A7 1903 28A9  ..)....
00B0 282F 30FF 1283 0085 3018 1683 0085 30FF  /.../...
00B8 1283 0086 1683 0186 30FF 0081 30FF 1283  ..)....
00C0 0087 1683 0187 30FF 1283 0088 1683 0188  ..)....
00C8 1209 0008 1283 00A9 3001 00A8 3008 0229  ..)....
00D0 1803 28DF 0829 1903 28DB 00A8 1803 1283  ..)....
00D8 00A8 08A8 28D6 09A8 1283 082A 0486 3008  ..)....*..
00E0 1283 0229 1C03 28F7 3010 0229 1803 28F7  ..)....)..
00E8 3008 02A9 0829 1903 28F3 00A8 1803 1283  ..)....
00F0 00A8 08A8 28EE 09A8 1283 082A 0487 3010  ..)....*..
00F8 1283 0229 1C03 290F 3018 0229 1803 290F  ..)....)..
0100 3010 02A9 0829 1903 2908 00A8 1003 1283  ..)....
0108 00A8 08A8 2906 09A8 1283 082A 0488 0008  ..)....*..
0110 1283 00A9 3001 00A8 3008 0229 1803 2924  ..)....).$.
0118 0829 1903 2921 00A8 1003 1283 00A8 08A8  ..)....).$.

```



Finalement, l’option ‘Verify’ permet de vérifier si le contenu de la ROM est identique au code chargé par le logiciel MpLab.