Fonctionnement carte I/O (draft)

La carte Entrées/Sorties se branche sur le port USB d'un PC.
Elle apparaît alors sous la forme d'un port série supplémentaire, ex: COM3 sous Windows, ou /dev/ttyUSB0 sous linux.
Ceci est vrai si la petite rom série 93C46 est absente, auquel cas le bridge USB/serie FTDI se déclare en tant que tel, avec un vendor/device ID USB adéquat.
Si la ROM est présente (et contient des ID propre au clubelek) alors la carte n'apparait plus comme un port série, et il faut l'attaquer avec un soft maison, ou modifier les parametres du driver FTDI pour utiliser ces nouveaux ID (fdti.inf? Sous windows, et /etc/hotplug/... sous linux?)

Dip switches :

Les deux blocs de DIP switches sur la carte servent a connecter/deconnecter des resistances de pullup sur les deux borniers correspondant.
Quand on veut utiliser un bornier en sortie, il faut deconnecter TOUS les pullups (positions OFF); pour l'utiliser en entrées, il faut connecter TOUS les pullup (position ON).
Evidement, pour utiliser telle broche en entrée, telle autre en sortie dans un même bornier, il faut positionner les pullup en respectant les règles ci dessus.

Boutons poussoir:

Celui-ci sert à faire un RESET hard de la carte.
Hors au RESET, la carte active le moniteur (boot loader) qui attend quelques secondes un caractere sur la liaison serie (cad, USB) du PC.
Si un caractere est recu (ex: on presse une touche dans l'emulateur de terminal) alors le boot loader rentre dans un petit menu interactif :
U pour Uploader du code executable dans le PIC
Q pour quitter le moniteur et executer l'appli

Si aucun caractere n'est recu au boot, le moniteur passe la main a l'appli.

Trames séries

Le principe de communication Carte/PC est le suivant:

Quand un des interlocuteur a quelquechose a dire, il l'envoie sur la liaison serie, qui est en fait véhiculée au dessus du lien USB, le tout de facon transparente.

En principe, le PC est Maître. La carte est esclave. Donc à priori la carte ne prend jamais la parole d'elle même.

Les trames échangées sont de la forme: (en bytes)
* commande =description commande
    - parametre1 parametre2 ...

Reponse: valeur

La commande est obligatoire.
Les parametres sont optionnels, et dépendent de la commande.
La réponse à la trame est optionnelle et dépend de la commande (par défaut il n'y a pas de réponse requise).

Card->PC:
80=interrupt
    - pin number

EE=unknown command or transmission error

PC->Card:
* 00 = analog configuration
  + configuration: 00 = No pin in analog mode
    01 = Pin 0 in analog mode
    02 = Pins 0, 1 and 3 in analog mode
    03 = Pins 0 to 4 in analog mode
    04 = Pins 0 to 5 in analog mode
    05 = Pins 0 to 7 in analog mode

* 01 = configure pin
  + pin number
  + pin configuration:
    00 = digital input
    01 = digital input with interrupt on rising edge
    02 = digital input with interrupt on falling edge
    03 = digital input with interrupt on both edges
    04 = digital output
    05 = servo ouput

  Answer frame: 01

* 02 = set pin (for a pin configured in digital output)
  + pin number
  + valeur : 0 = logical 0
    other value = logical 1

  Answer frame: 02

* 03 = read digital pin

(for a pin configured in digital input)
+ pin number
Answer frame: 03 + pin state (00 or 01)

* 04 = read analog pin (for a pin configured in analog input)
+ pin number
Answer frame: 04 + result on 2 bytes between 0 and 1024 (high byte first)

* 05 = set motor 1
+ direction: 00 = brake
01 = forward
02 = backward
+ one byte for speed (from 0 to 255)
Answer frame: 05

* 06 = set motor 2
+ direction: 00 = brake
01 = forward
02 = backward
+ one byte for speed (from 0 to 255)
Answer frame: 05

* 07 = set servo (for a pin configured in servo mode)
+ pin number
+ value on 2 bytes between 0 and 9999 (high byte first)
Answer frame: 06

* 08 = set servo speed (for a pin configured in servo mode)
+ pin number
+ speed on 1 byte between 0 and 255
(0 is no speed control, 1 = min speed, 255 = max speed)
Answer frame: 7

* 09 = set the red LED
+ value : 0 = off
other value = on
Answer frame: 09

* 0A = set the green LED
+ value : 0 = off
other value = on
Answer frame: 0A

* AA = test if card is present
Answer frame: AA

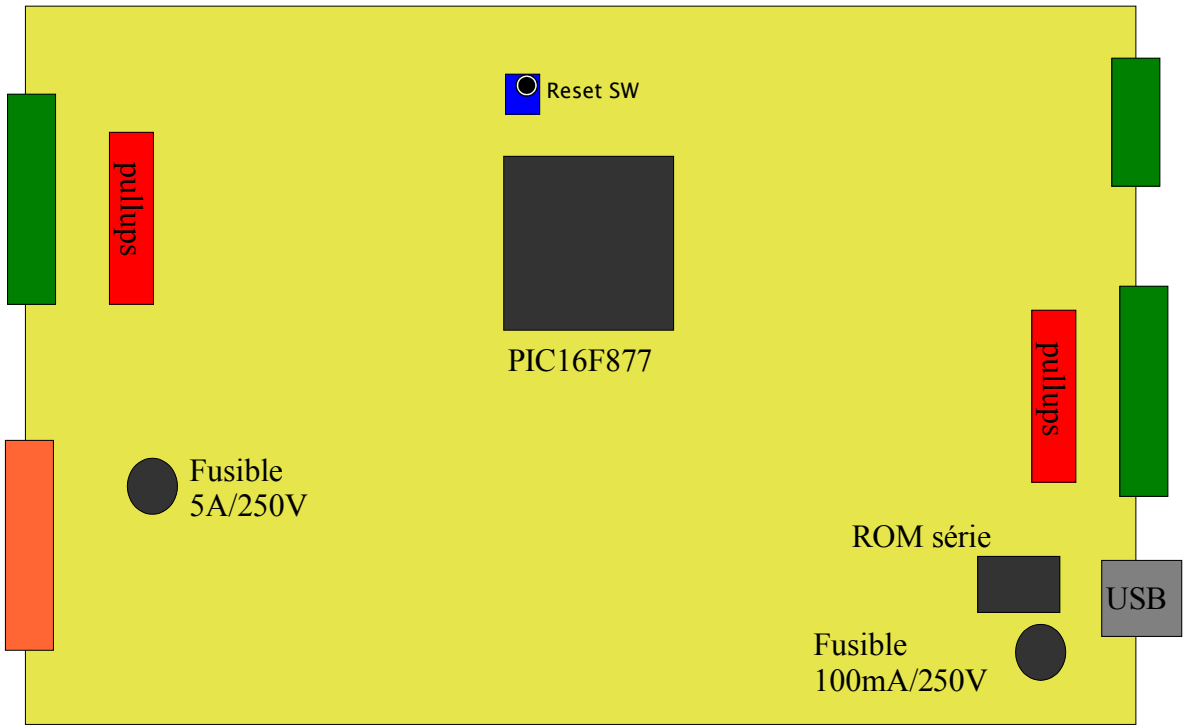
* FF = card reset
Answer frame: FF

Exemple:

```
FF=>            // reset card
=>FF            // card succesfully reset
09 01=>         // set red led
=> 09
....etc.....
```

(evidemment, il ne faut pas mettre les '=>' - de toute facons, ces commandes sont en hexadecimal !!!!!)

Schéma de la carte d'entrees/sorties



Carte Entrees/sorties

API carte Entrees/Sorties:

01+PIN+00 -> 01 => set digit PIN to input
01+PIN+02 -> 01 => set digit PIN to triglow interrupt
01+PIN+04 -> 01 => set digit PIN to output
02+PIN+V -> 02 => write value V (=00 or 01) to digit PIN
03+PIN -> 03+V => read PIN value (=V)

API Carte I/O

```
#define CMD_CONFIG_ANA_PINS 0x00
#define ANA_NO_PIN 0x00 // No pin in analog mode
#define ANA_1_PIN 0x01 // Pin 0 in analog mode
#define ANA_3_PINS 0x02 // Pins 0, 1 and 3 in analog mode
#define ANA_5_PINS 0x03 // Pins 0 to 4 in analog mode
#define ANA_6_PINS 0x04 // Pins 0 to 5 in analog mode
#define ANA_8_PINS 0x05 // Pins 0 to 7 in analog mode
#define CMD_CONFIG_PIN 0x01
#define PIN_INPUT_MODE 0x00 // digital input
#define PIN_INPUT_RE_INT_MODE 0x01 // digital input with interrupt on rising edge
#define PIN_INPUT_FE_INT_MODE 0x02 // digital input with interrupt on falling edge
#define PIN_INPUT_BE_INT_MODE 0x03 // digital input with interrupt on both edges
#define PIN_OUTPUT_MODE 0x04 // digital output
#define PIN_SERVO_MODE 0x05 // servo ouput
#define PIN_ANA_MODE 0x06 // analog input
#define CMD_SET_PIN 0x02
#define CMD_GET_DIG_PIN 0x03
#define CMD_GET_ANA_PIN 0x04
#define CMD_SET_MOTOR1 0x05
#define CMD_SET_MOTOR2 0x06
#define MOTOR_BRAKE 0x00
#define MOTOR_FORWARD 0x01
#define MOTOR_BACKWARD 0x02
#define CMD_SET_SERVO 0x07
#define CMD_SET_SERVO_SPEED 0x08
#define CMD_SET_RED_LED 0x09
```

```
#define CMD_SET_GREEN_LED 0x0a
#define CMD_PRESENT 0xaa
#define CMD_INTERRUPT 0x80
#define CMD_ERROR 0xee
#define CMD_RESET 0xff
```