Protocole TODO

# Introduction

C’est un protocole simple qui permet l’envoi en I²C de messages. Il permet à l’expéditeur de s’authentifié, ainsi de spécifié des options indiquant la présence ou non de cryptage par exemple.

## Motivations :

## Principes :

* Couche réseau se situant au dessus de la couche liaison (I²C).
* Non connecté, parce que l’I²C ne l’est pas, et que les cartes ne sont pas assez puissantes et qu’elles ont peu de RAM.
* Une en-tête variable, ce qui permet l’évolution du protocole en assurant la rétrocompatibilité.

# Configuration et utilisation I²C

## Configuration :

* Adresse (unique sur le réseau)
* Adresse sur 7bits
* Fréquence
* Rapport cyclique
* Type d’acquittement
* InputClockFrequencyMHz

## Utilisation :

Dans le cas ou la carte doit utiliser l’I²C pour une raison quelconque sans passé par le protocole TODO, Il y a possibilité de figer le protocole TODO. Il faut pour cela appelé la fonction TODO\_freeze(u8). Si le protocole est freezé, il appellera une autre fonction lors d’une interruption I²C. C’est au développeur de définir cette fonction.

Conseil : Il est fortement conseillé d’être le maître I²C avant de freeze.

# En-tête

Les données échangées posséderont l’en-tête suivante.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit7** | **Bit6** | **Bit5** | **Bit4** | **Bit3** | **Bit2** | **Bit1** | **Bit0** |
| Adresse source (expéditeur) | | | | | | | 1 |
| TY | | CR |  |  |  |  | MO |
|  |  |  |  |  |  |  | *MO* |
| Datas (fini par 0x00) | | | | | | | |
|  |  |  |  |  |  |  |  |

* TY : définie le type de l’entité :
  + 00 : Terminal
  + 01 : Serveur
  + 10 : Périphérique
  + 11 : Autre
* CR : Utilisation du cryptage :
  + 0 : non
  + 1 : oui
* MO : More Option :
  + 0 : Les octets suivants sont les Datas
  + 1 : Il y a au moins un octet supplémentaire d’option.
* Datas : Donné contenu par la trame : terminé par un octet 0x00.

# Gestion des erreurs :

* Si une entité A s’adresse à une autre entité B en mode lecture alors B flood A jusqu’à la fin de la communication. (interrupt ADDRESSSENTMATCHED).
* Si les données ne se terminent pas par la caractère nul, alors le paquet est ignoré.

# Interface pour l’applicatif :

* init(Type, uint8 addr, uint CRYPT\_KEY) :
* aux\_interrupt( (voi\*)(Func\*)(void) ) :
* recv() : (interrupt RXNOTEMPTY ). bloquant, remplie un buffer statique
* send(uint8 addr, char\* data, uint16 size) :
* freeze(uint8) : unfreeze(0), freeze( !0 )
* close() :

# Ouverture :

## Gestion dynamique des adresses.

* Paquet un bit NO\_ADDRESSS demandera une adresse serveur.