# Le terminal I²C

## Introduction

Nous allons définir dans ce document le cahier des charges d’une application pouvant être utilisé comme sujet de mini-projet de CS260.

Cette application est utilisée dans un réseau I²C comportant des terminaux, des périphériques et des serveurs.

## Taxonomie

Périphérique - Composant électronique comportant un module I²C, ajouté grâce à un STM8 ou non. Exemples :

* Moteur contrôlé en PWM
* Matrice de LED
* Horloge temps réel ou RTC

Terminal - Périphérique d’entrée/sortie permettant une interaction entre le réseau I²C et un opérateur. Exemples :

* Clavier numérique contrôlé par un STM8,
* Hyperterminal communiquant avec un STM8 via une liaison RS232.

Opérateur - Personne physique agissant sur un terminal afin de provoquer une action sur un périphérique I²C.

## Présentation de l’environnement

Pour ce mini-projet, un réseau I²C a été créé. Celui-ci comporte au moins un serveur et un ou plusieurs périphériques. L’adresse du serveur est connue.

Le serveur embarqué sur un STM8 contient une CLI (Command Line Interface). Ainsi, ce STM8 est capable de recevoir une chaine de caractères via le bus I²C, de l’interpréter, puis d’agir en conséquence. Il peut par exemple lire l’heure sur une horloge temps réel telle que le DS1307 sur réception de la chaine « lireHeure ».

## But de l’application

Le but de ce mini-projet est d’intégrer un terminal à ce réseau. Le terminal comportera un clavier numérique 5x4 ainsi qu’un hyperterminal branché via l’UART du STM8. L’opérateur voulant se servir du terminal devra entrer un code d’accès sur le clavier numérique. Si ce code est valide, l’opérateur aura alors accès au clavier de l’hyperterminal. Il pourra y entrer des commandes à destination du serveur.

## Cahier des charges

1. A l’état de repos, l’hyperterminal est figé et seul le clavier numérique est accessible.
2. Tant que le code d’accès valide n’a pas été entré et validé par la touche ‘V’ sur le clavier numérique, le système reste en état de repos.
3. Lorsque le code d’accès valide est validé, l’hyperterminal devient alors accessible.
4. L’appui sur une touche du clavier numérique pourra être accompagné d’un bip sonore.
5. Un code d’accès invalide pourra déclencher un avertissement sonore.
6. A tout moment de la saisie du code d’accès, l’opérateur pourra le corriger grâce à la touche ‘C’ du clavier numérique.
7. Le code d’accès valide utilisé pour comparaison avec le code d’accès opérateur sera contenu dans l’EEPROM du STM8.
8. Après un déverrouillage, il est possible de verrouiller à nouveau l’hyperterminal une séquence de touches sur le clavier numérique.
9. Un temps d’inactivité sur l’hyperterminal pourra mener au verrouillage automatique de la session.
10. Il est possible de configurer un nouveau code d’accès après déverrouillage d’une session et entrée d’une certaine séquence de touches sur le clavier numérique.
11. Si une commande comporte plus de 32 caractères, alors les caractères excessifs seront ignorés.
12. L’envoi de la commande au serveur ne se fait qu’après appui sur la touche « entrée » de l’hyperterminal.
13. Les données transmises au serveur pourront être chiffrées à l’aide de la clef commune à tous les serveurs et terminaux du réseau I²C utilisant le chiffrement des données.

## Implémentation sur un STM8S :

Nous implémenterons le protocole à la vitesse de communication standard de l’I²C, 100kHz. Premièrement, il faut mettre le bit F/S de I2C\_CCRH à 0. Nous voulons ensuite une période de 10µs. La période de Signal CLock (SCL) vaut, en mode standard, 2\*CCR\*.

Ainsi pour un registre FREQR de 16MHz (FREQR = 0x10) nous aurons . Il faut donc placer Clock Control Register (CCR) à 80 (soit 0x50) pour avoir .

A la vitesse de communication standard, le rapport cyclique n’a pas d’impact. Le bit DUTY\_CYCLE du I2C\_CCRH sera donc arbitrairement mis à 0.

L’acquittement se fait sur l’octet courant. Ainsi le bit ACK de I2C\_CR2 sera positionné tandis que le bit POS de I2C\_CR2 sera à 0.

Le protocole doit laisser au développeur la liberté de communiquer en I²C sans utiliser des paquets TODO.

* init(uint8 addr, uint8 type, uint8 cryptKey)
* uint8 send(const void \*msg, size\_t len, int flags);
* int recv(int s, void \*buf, int len, unsigned int flags);