Thème n°2 : Interfaces de communication (1. UART/USART)

CELECT7 : Applications des microcontrôleurs

M. Zwingelstein

rev. 2020

Travail préparatoire

Lectures de la partie préparatoire :

Lire attentivement :

1. Datasheet de l'ATmega328p : [lien]
   - paragraphe 19.1 (Features)
   - paragraphe 19.2 (Overview)
   - paragraphe 19.4 (Frame formats)
2. Abstraction arduino pour la liaison série :
   - basée sur l'interface série du microcontrôleur (hard): [lien]
   - entièrement implémentée en soft : [lien]

Parcourir :

3. Datasheet de l'UART sans fil HC-11 : [lien]

Questionnaire de la partie préparatoire :

1. Architecture ATmega 328p
   - Que signifient les acronymes UART et USART?
   - Quels sont les noms des registres qui permettent de configurer l'USART/UART du microcontrôleur ATmega 328p ?
     - registre pour contrôler :
     - registre pour voir l'état :
     - registre pour choisir le débit :
     - registre pour choisir le format de la trame :
   - Que permet un circuit USART que ne permet pas un circuit UART ?
   - En vous appuyant sur le paragraphe 19.6 et le tableau 19-12 de la datasheet, déterminer quel doivent être les contenus des registres **UBRR0H** et **UBRR0L** afin de paramétrer la liaison série à 57600 bit/s si la fréquence d'horloge du microcontrôleur est de 16 MHz.
2. Ecrire la structure de la trame série correspondant à la transmission de la valeur décimale 1 avec une parité paire, et 1 bit de stop.
3. Abstraction arduino : en vous inspirant des exemples sur `arduino.cc`, écrire un sketch arduino qui reçoit une valeur entrée au clavier depuis le terminal série côté ordinateur, et qui affiche sur le terminal le sigle `OK` suivi d'une tabulation et de la valeur saisie au clavier.

Organisation du thème

- Partie 1 :
  - Observation et analyse des signaux physiques sur une liaison série. Mise en rapport avec les fonctions arduino `write()` et `print()`
  - Lab 1
- Partie 2 :
  - Utilisation directe des registres de l'UART de l'AVR 328p, en lieu et place des fonctions arduino pour gagner en taille de programme
  - Lab 2
- Partie 3 :
  - Mise en oeuvre du circuit HC11 basé sur la puce radio fréquence CC1101 ; paramétrage et communication de données
  - Lab 3

Cours 1: Liaison série UART

Notion de bus

Une *interface* (ou *bus*) de communication série permet de faire transiter des informations binaire les unes derrière les autres, entre deux points.

Un microcontrôleur comporte généralement plusieurs types d'interface de communication série :

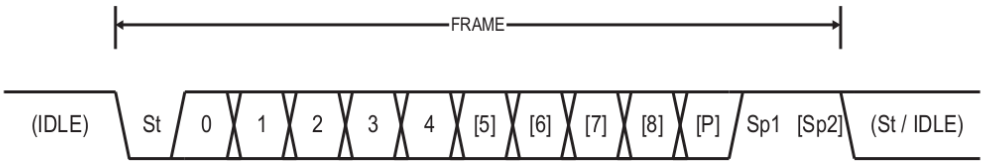
- UART/USART, I2C, SPI, USB, 1-wire…
- Celles-ci lui permettent de s'interfacer avec d'autres périphériques ou systèmes.
- L'UART est l'interface de communication série *la plus simple* et la *plus répandue*.

Caractéristiques principales de l'UART

UART : Universal Asynchronous Receiver Transmitter

- Asynchrone (pas de signal d'horloge)
- Liaison point à point bi-directionnelle, full-duplex
- 3 fils suffisent : TX, RX, GND
- Débits : 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200 bits/s

Format de trame



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

- Emetteur et récepteur doivent utiliser le même format de trame
- Calcul du bit de parité :

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

$\mathbf{P_{even}}$	Parity bit using even parity
$\mathbf{P_{odd}}$	Parity bit using odd parity
$\mathbf{d_n}$	Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

Abstraction arduino associée à l'UART

Relire si besoin la documentation fournie sur arduino.cc (voir travail préparatoire)

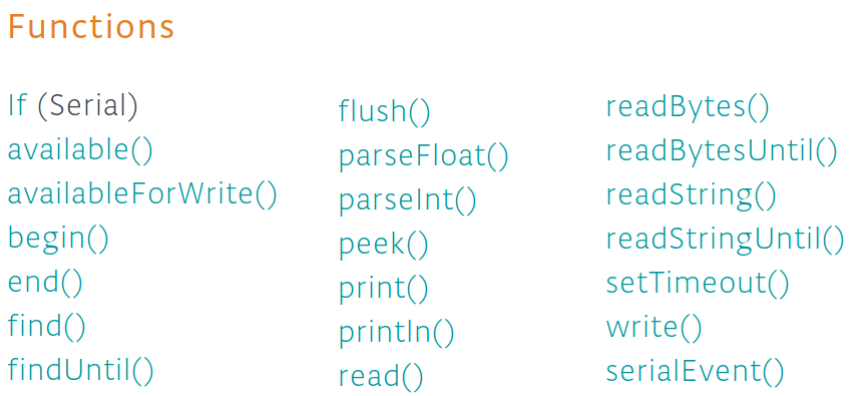


Figure 1 – Fonctions arduino associées à la liaison série

Pour rappel :

— la fonction `write()` permet d'écrire directement *un octet* sur le port série
— la fonction `print()` envoie l'un après l'autre les *caractères ascii* associés aux caractères en argument.

Lab 1: Mise en oeuvre de la liaison série UART en utilisant l'abstraction arduino

Objectif et matériel utilisé

- L'objectif de ce premier lab est de bien comprendre le lien entre les fonctions arduino et la forme du signal électrique qui est réellement véhiculé sur la liaison série
- Matériel utilisé : oscilloscope, carte arduino uno

L'ensemble des tests de ce Lab 1. devront être reportés dans un tableau

Entrées du tableau :

- fonction arduino testée
- nombre d'octets envoyés
- dessin de la trame observée à l'oscilloscope
- texte explicatif sur la trame observée

Travail à effectuer :

1. Ecrire un sketch arduino qui envoie sur le port série, à intervalles de temps réguliers, l'octet correspondant au chiffre 1 (`Serial.write(1);`). La liaison série devra être paramétrée comme suit :
   - débit : 57600 bits/s
   - pas de bit de parité (mode par défaut)
   - 1 bit de stop (mode par défaut)

   Penser à mettre un `delay` à la fin du `loop()` afin que les envois successifs soient espacés d'environ 1 seconde, ce qui permettra une interprétation plus facile du signal observé.

   Observer le signal généré en branchant la sonde de l'oscilloscope sur la broche TX de l'arduino.

   Dessiner le signal (trame série) et expliquer sa forme.
2. Reprendre le point 1., mais avec les configurations suivantes :
   - 2 bits de stop, pas de bit de parité
   - 1 bit de stop, parité paire
   - 1 bit de stop, parité impaire
3. Reprendre le point 1., mais en envoyant les données suivantes au lieu de la valeur 1 (s'inspirer du code page suivante)
   - un **`byte`** de valeur 2, avec la fonction `Serial.write()`
   - un **`byte`** de valeur `'a'`, avec la fonction `Serial.write()`

- un **byte** de valeur 2 avec la fonction `Serial.print()`
- un **byte** de valeur `'a'` avec la fonction `Serial.print()` <– Ré : 2 octets, 10011100 (0x39 à l'envers, qui est le code ascii de 9) suivi de 11101001 (0x37 à l'envers, qui est le code ascii de 7 –>)
- un **byte** de valeur `'a'` avec la fonction `Serial.print()` avec comme second argument (à tester l'un après l'autre) :
  - `BIN`
  - `DEC`
  - `HEX`

Pour tous ces test, vous observerez à l'oscilloscope le nombre d'octets transmis par chaque commande. Ce nombre est à comparer avec la valeur retournée par la fonction `Serial.write()` ou `Serial.print()`.

Exemple de fonction `loop()` pour réaliser le premier test de la question 3 :

```
1 loop() {
2     byte valeur = 2;
3     int nbBytes = Serial.write(valeur);
4     Serial.print('\t');
5     Serial.println(nbBytes);
6     delay(1000);
7 }
```

Cours 2: Mise en oeuvre de la liaison série dans l'ATmega 328p

Diagramme en bloc

Figure 20-1. USART Block Diagram(1)

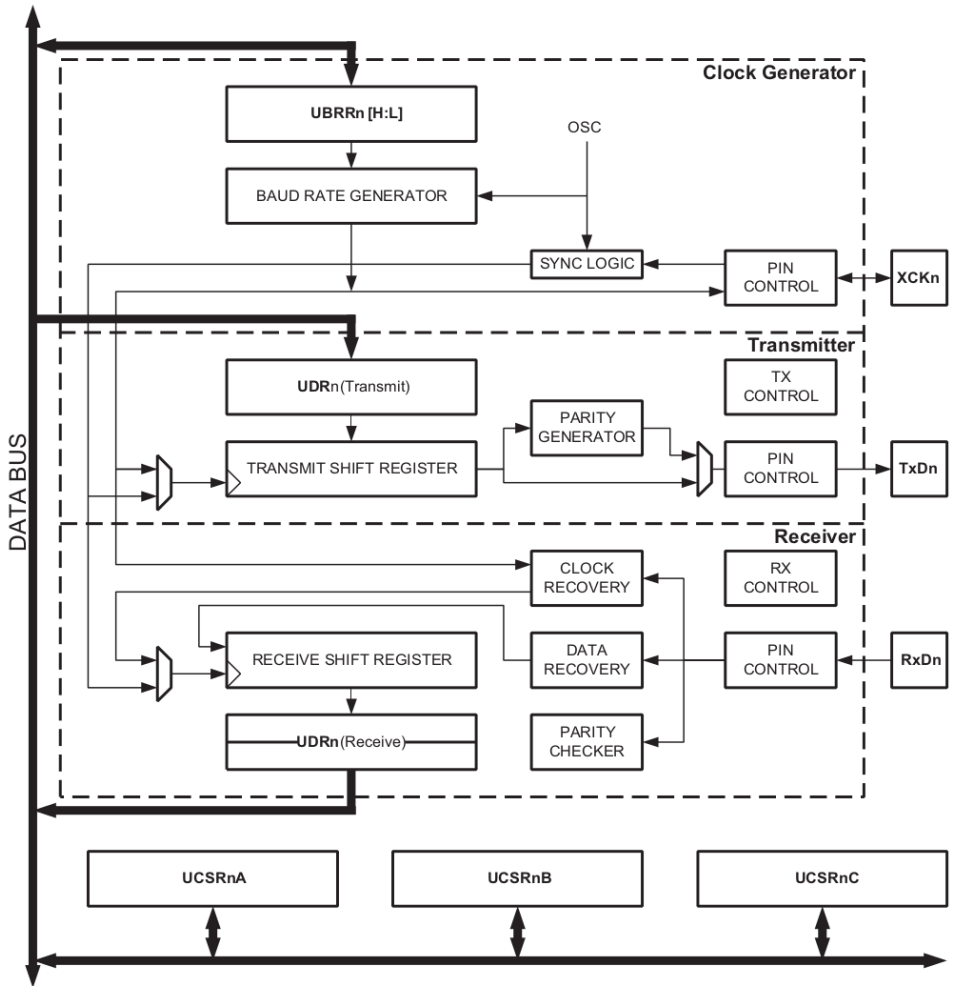


FIGURE 2 – source : datasheet AVR328p

Registres

5 registres, répartis en trois catégories :

- Un registre des données émises et reçues : **U**SART I/O **D**ata **R**egister (**UDR0**, 8 bits)
- Trois registres de contrôles et de status : **U**SART **C**ontrol and **S**tatus **R**egister (3 registres de 8 bits, **UCSR0A**, **UCSR0B**, **UCSR0C**)

— Unregistre de contrôle du débit : **U**SART **B**aud **R**ate **R**egister (**UBRR0H** pour les 8 bits de poids fort et **UBRR0L** pour les 8 bits de poids faible)

Mode d'emploi

Configuration du débit

— Agir sur les registres **UBRR0H** et **UBRR0L** (tableaux 19-9 à 19-12 de la datasheet)

Configuration de l'autorisation d'envoi/réception de données

— Agir sur les bits TXEN0/RXEN0 du registre **UCSR0B** (paragraphe 19.10.3 de la datasheet)

Transmission d'un caractère

— En écrivant simplement sa valeur dans le registre **UDR0**. Voir la datasheet sections 19.6 et 19.7, où l'on apprend :
  — qu'une valeur écrite dans le registre **UDR0** sera envoyée dans le registre d'émission et donc émise
  — qu'une valeur reçue dans le registre de réception sera obtenue en lisant le contenu du registre **UDR0**

Lab 2. Programmation directe des registres de l'UART

Programme préliminaire

(utilisant l'abstraction arduino)

1. Reprendre le programme `seance1_a1.ino` de la séance 1. Remplacer la ligne 24 par `Serial.println(inputs,BIN);` puis compiler et téléverser.
2. Prendre un fil mal-mal, relier un côté à la masse, et l'autre à l'une des E/S arduino n°2 à n°9 (alternativement) et vérifier que vous obtenez l'affichage attendu.
3. Remplacer à présent la ligne 24 par `Serial.write(inputs);` puis observer à l'oscilloscope le signal sur la liaison série, tout en continuant de relier une des E/S arduino n°2 à n°9 à la masse.

   Justifier vos observations.
   - noter la taille du programme compilé
   - enregistrer le programme sous le nom `seance2_lab2_1.ino`

Réduction de la taille du programme préliminaire

On cherche à réduire la taille du programme précédent, en configurant directement les registres associés à la l'UART au lieu de passer par la couche d'abstraction arduino.

Pour rappel, nous avons vu à la séance 1 comment réduire la taille du programme pour tout ce qui concerne la gestion des E/S.

4. Enregistrer le programme précédent sous le nom `seance2_lab2_2.ino`
5. Reprendre du travail préparatoire la configuration des registres **UBRR0H** et **UBRR0L** pour avoir un débit de 57600 bits/s, sachant que la fréquence de l'oscillateur est de 16 MHz
   - `UBBR0H = 0B...` (compléter)
   - `UBBR0L = 0B...` (compléter)

   On précise que par défaut, ces deux registres sont à zéro.
6. Les registres **UCSR0A** et **UCSR0C** seront laissés à leurs valeurs par défaut.

   Par contre, le registre **UCSR0B** contient le bit TXEN0 qui doit être mis à 1 pour configurer l'autorisation de transmission.
   - Compléter la ligne de code utilisant un masquage *OU* afin de mettre à 1 le bit TXEN0, sans modifier les autres bits du registre **UCSR0B** :
     - `UCSR0B |=...` (compléter)

7. Dans le programme `seance2_lab2_2.ino`, commenter la ligne `Serial.begin(57600);` et la remplacer par les configurations des registres vues précédemment. Compiler, téléverser et tester le code, pour vérifier qu'il fonctionne toujours correctement
8. Dans le programme `seance2_lab2_2.ino`, commenter la ligne `Serial.println(inputs, BIN);` et la remplacer par une écriture directe de l'octet `inputs` dans le registre **UDR0**
9. Compiler, téléverser et tester le code, pour vérifier qu'il fonctionne toujours correctement. Noter la taille du programme compilé et apprécier la réduction obtenue
10. Afin de diminuer encore la taille du programme, apporter les mêmes modifications qu'en séance 1 (Lab, partie 1). Vous nommerez le programme final obtenu `seance2_lab2_3.ino`

    Compiler, téléverser et tester le programme. A quelle taille de programme aboutissez vous ?

Cours 3: Exemple d'utilisation d'un périphérique série

UART sans fil HC-11

Le circuit HC-11 est un périphérique de type *UART sans fil* fabriqué par la societé Chinoise Wavesen. Il est composé d'un module radio fréquence et d'un micro-contrôleur.

Le rôle du micro-contrôleur est de convertir l'interface du composant radio fréquence en une interface UART.

- Avantage : grande facilité de mise en oeuvre d'une liaison sans fil point à point
- Inconvénient : l'UART sans fil ne permet généralement pas d'exploiter toutes les possibilités du circuit radio fréquence (ex : type de modulation, débits…)

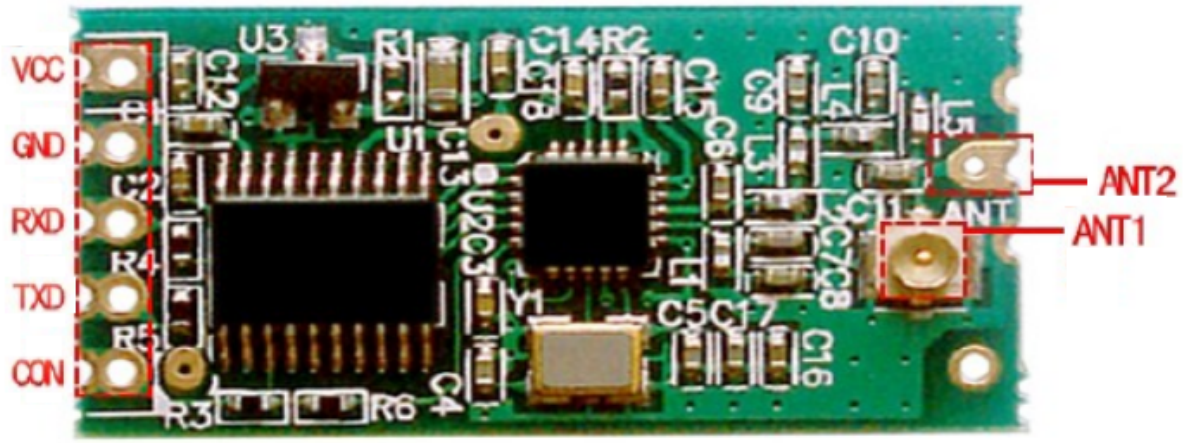


Figure 3 – Circuit HC11

Le circuit HC-11 est basé sur les composants très répandus suivants

- Texas Instrument CC1101 (composant radio fréquence)
- Motorola MC68HC11 (microcontrôleur)

Il existe un autre modèle d'UART sans fil, le HC12, plus récent et basé sur les composants suivants :

- un composant radio fréquence Silicon Labs Si4463
- un microcontrôleur STMicroelectronics STM8S003F3

Les fonctionnements du HC-11 et du HC-12 sont très similaires.

Principe d'utilisation

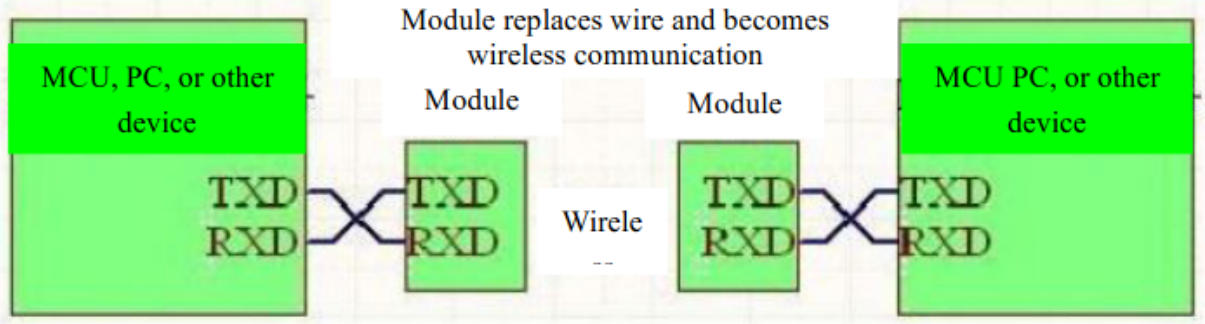


FIGURE 4 – Principe d'utilisation d'une UART sans fil

Modes de configuration du HC-11

Le circuit HC-11 peut être configuré selon 4 modes correspondant à des distances de transmission, des débits et des consommations différents :

Mode	FU1	FU2	FU3	FU4	Remark
Idle current	3.5mA	80μA	22mA	22mA	Average value
Transmission time delay	20mS	380mS	2mS	7 mS	Sending one byte
Loopback test time delay 1	31mS	8mS	22mS		Serial port baud rate 9,600, sending one byte
Loopback test time delay 2	31mS	18mS	40mS		Serial port baud rate 9,600, sending ten bytes

FIGURE 5 – FU1: normal, FU2: basse consommation, FU3: faible latence, FU4: longue portée.

Composant radio fréquence TI CC1101

Le circuit CC1101 de Texas Instrument est un *transceiver* (contraction de *transmitter-receiver*) RF à bande étroite qui opère dans la bande ISM des 433 MHz.

Caractéristiques principales :

— basse consommation (30 mA maxi en Tx, sous 3,3 V. 10 nA en mode sleep)

— sensibilité élevée (–116 dBm à 0.6 kBaud pour un PER - Packet Error Rate - de 1%)
— modulation numérique paramétrable (OOK, ASK, FSK, MSK)
— débit paramétrable, maxi 500 kBauds
— puissane maximale émise +12 dBm. Portée x100m en ligne directe

Ce composant n'implémente pas de fonctionnalité réseau (se limite à la couche 2 du modèle OSI).

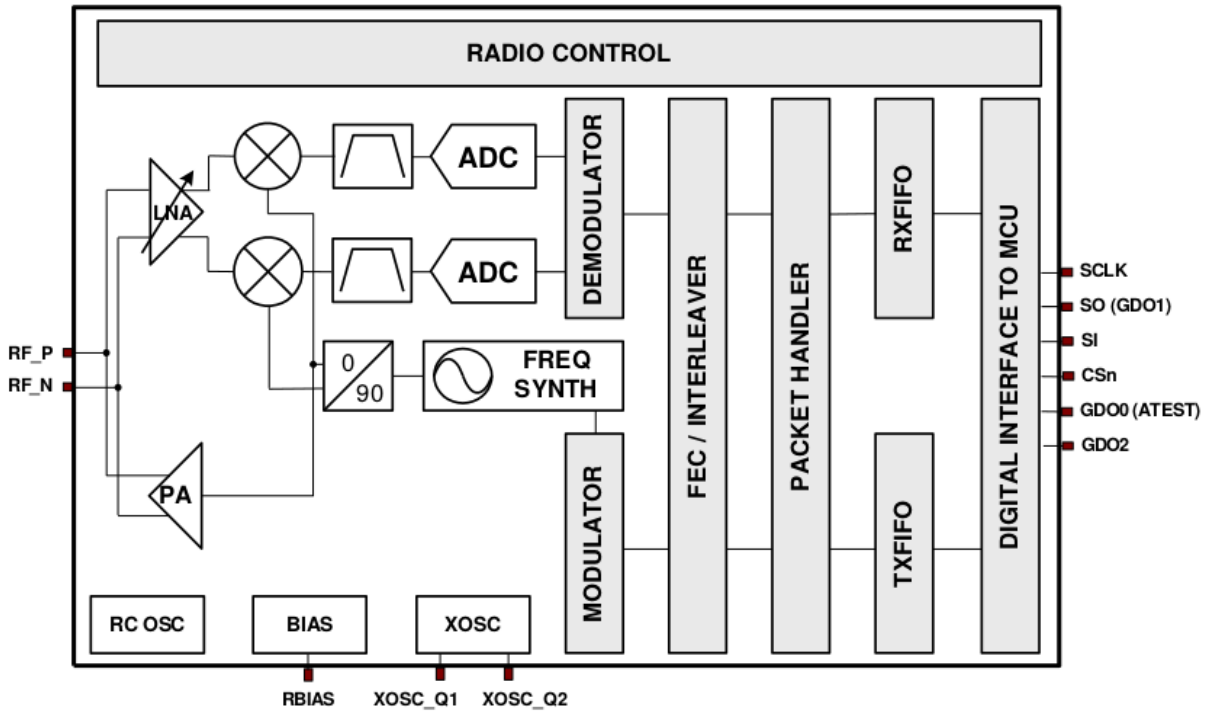


Figure 6 – Diagramme en block simplifié du CC1101

8 Configuration Overview

CC1101 can be configured to achieve optimum performance for many different applications. Configuration is done using the SPI interface. See Section 10 below for more description of the SPI interface. The following key parameters can be programmed:

- Power-down / power up mode
- Crystal oscillator power-up / power-down
- Receive / transmit mode
- RF channel selection
- Data rate
- Modulation format
- RX channel filter bandwidth
- RF output power
- Data buffering with separate 64-byte receive and transmit FIFOs
- Packet radio hardware support
- Forward Error Correction (FEC) with interleaving
- Data whitening
- Wake-On-Radio (WOR)

Details of each configuration register can be found in Section 29, starting on page 66.

Figure 13 shows a simplified state diagram that explains the main **CC1101** states together with typical usage and current consumption. For detailed information on controlling the

FIGURE 7 – Possibilité de configuration du CC1101

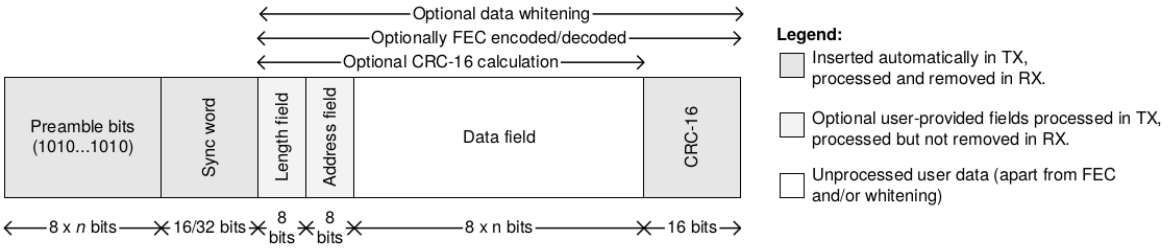


Figure 19: Packet Format

FIGURE 8 – Format de trame transmise avec le CC1101

Lab 3. Mise en oeuvre du circuit de communication HC-11

Branchement

HC-11	Arduino uno
VCC	3.3 V ou 5 V
GND	GND
RXD	3 (TX)
TXD	2 (RX)
SET	GND

Le microcontrôleur ATmega 328p de la carte arduino uno ne contient **qu'un seul UART**, et nous l'utilisons déjà pour dialoguer entre l'arduino et le terminal série sur le PC (via le câble USB). Comment faire ?

Heureusement, il est toujours possible d'implémenter **de manière logicielle** un deuxième UART. Ceci peut être fait facilement avec la bibliothèque arduino `SoftwareSerial`, qui sera paramétrée afin que le TX se trouve sur la broche 3, et le RX sur la broche 2.

Configuration du HC11 : Mode _commande

Ce mode est utilisé pour **configurer** le module HC-11 (voir la datasheet du module [lien]). Il sera utilisé comme suit :

— On communique à arduino une commande AT (ex : AT+V) depuis le terminal série du PC (en hard : UART de l'ATmega 328p)
— Le microcontrôleur ATmega 328p lit cette commande et la retransmet au HC-11 via une deuxième liaison série (en soft)
— Le tranceiver HC-11 répond à cette commande (via la liaison série soft HC-11/ATmega328p)
— Cette réponse est affichée sur le terminal série de l'IDE arduino (via la liaison série hard ATmega 328p/PC)

1. Compiler et téléverser le programme fourni `seance2_lab3_1.ino` dans l'ATmega 328p. La deuxième liaison liaison série, entre l'ATmega 328p et le HC-11 est configurée pour avoir RX sur la broche arduino n°2 et TX sur la broche arduino n°3

2. En utilisant le terminal série de l'IDE arduino, et en vous basant sur la datasheet du module HC-11, réaliser les opérations suivantes :
   — afficher la version du firmware de l'HC-11
   — afficher les informations de mode, de débit de la liaision série, de canal radio, d'adresse du module HC-11 et de puissance RF émise
   — programmer le module HC-11 afin que la puissance émise soit de 1 mW

   Pour chaque opération, noter la commande utilisée ainsi que le résultat affiché.

Mode *transparent*

Le mode transparent est utilisé pour **communiquer** sans fil entre deux modules HC-11.

Pour la suite de ce lab, travailler en groupe de deux binômes afin d'utiliser deux cartes arduino associées chacune à un module HC-11.

Ceux-ci devront porter la même adresse et utiliser le même canal radio.

— Organisez-vous afin que deux canaux radio identiques ne soient pas utilisés deux fois dans la salle, sous peine d'interférences !
— Utilisez la méthode du point n°1 afin de configurer l'adresse des modules et la bande radio utilisée.

3. Pour communiquer en mode transparent, il suffit maintenant de déconnecter de la masse la broche SET des modules HC-11 (on laissera cette broche flottante).
   — reprendre le sketch `seance2_lab3_1.ino` et l'enregistrer sous le nom `seance2_lab3_2.ino`
   — modifier le sketch en remplaçant la fonction `read()` par `readString()` et `write()` par `println()`
   — compiler et téléverser le programme dans chacun des modules HC-11 puis tester la communication
   — commenter vos résultats

Mode *Wireless IO*

4. Inspirez-vous de la documentation du HC 11 (sections V et VI) afin de réaliser la manipulation suivante :

   Un module HC-11 doit piloter à distance l'état d'une LED RVB connectée aux broches 3, 4 et 5 d'un autre module HC-11. Le pilotage doit se faire en changeant l'état des broches 3, 4 ou 5 du premier module HC-11.

Expliquer votre démarche.