ulco Université Littoral Côte d'Opale

Examen de Systèmes Embarqués
partie Microcontrôleur

EILCO, ING2
Semestre 1
P.-A. Hébert
18/12/2024

Consignes

- Sans document. Pas de calculatrice (car inutile : les calculs sont simples).
- Durée : 1h.
- Énoncé : 1 page de présentation + 2 pages de questions + 5 pages de documentation (timers).
- Barême : 6 questions sur 20 points
- Codez l'ATmega328p, en langage Arduino (C/C++).
- Manipulation des ports : **uniquement à l'aide des registres.**
- **Justifiez tous les calculs**
- Avant de répondre à une question, lisez-la entièrement.
- **En cas de souci, supposez la question résolue, et passez à la suivante.**

Communication série asynchrone

On souhaite envoyer un mot 8 bits de manière sérielle, c'est-à-dire bit par bit, à la vitesse de 16 000 bits par seconde. La communication (émission seulement) se fera sur une seule ligne, donc sans synchronisation par signal d'horloge (c'est une liaison série *asynchrone*).

Protocole de communication asynchrone

Afin de se synchroniser, le circuit récepteur (non traité ici) devra cadencer sa lecture des bits à la même vitesse. Mais pour l'avertir du moment où débute la transmission du mot, notre émetteur Atmega328p devra respecter le protocole suivant :

- Mise à 1 (état haut) de la ligne de communication, dans **l'attente du début de la transmission.**
- Envoi de 2 bits à 0 (état bas) : au **démarrage** de la transmission du mot, pour avertir le récepteur. On les nomme « **bits start** ».
- Envoi des 8 bits du mot : successivement, du bit de poids **faible** au bit de poids **fort.**
- Envoi d'1 bit à 1 : pour remettre la ligne de communication à son état initial d'attente. On le nomme « **bit stop** ». C'est la **fin** de la transmission du mot.

Sur la figure 1, on observe l'évolution de l'état de la ligne de communication lors de la transmission du mot `B11001000` (dont les bits sont lus de droite à gauche), qui démarre à $t = 4$ et finit à $t = 15$.

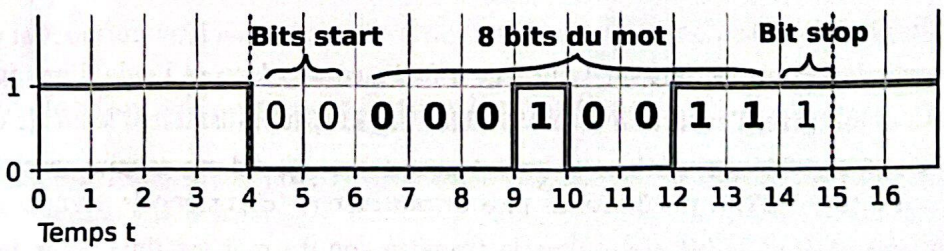


FIGURE 1 – Transmission du mot `B11001000`. Les 2 lignes verticales en pointillés indiquent le début et la fin de l'opération. Sont donc transmis : 2 bits start + 8 bits de mot + 1 bit stop.

Partie 1 - Transmission (émission) par timer

C'est le **timer 0** qui gérera cette tâche : il déclenchera l'envoi de chaque bit, à la vitesse (=fréquence) de 16 000 bits/seconde.

Plus précisément, il enverra 16 000 **demandes d'interruption** par seconde **en permanence**, mais on jouera sur **l'autorisation de cette interruption**, pour que le processeur n'accepte les demandes que lorsqu'une transmission de mot est en cours.

Attention, le timer **ne produit pas lui-même** le signal : ce sera le rôle du processeur, par manipulation d'un port parallèle, sur la broche 8 (cf. Q3). Le timer se contente d'indiquer au processeur quand doit se réaliser la transmission d'un bit.

Q 1. Écrire la fonction `void init_T0()` qui configure le **timer 0** dans le mode de comptage - vu en cours - qui réduit aux maximum l'écart entre durée de cycle souhaitée, et durée de cycle obtenue. *(6 points)*

Vous respecterez les contraintes suivantes :

- Le **prédiviseur** doit être optimisé pour que la fréquence d'interruption obtenue soit aussi proche que possible de celle souhaitée.
- **Ne pas autoriser** l'interruption souhaitée (elle le sera lorsqu'il y aura un mot à transmettre).
- **Démarrer** le compteur (le timer compte aussitôt).
- Pour rappel, sur l'Arduino Uno, **Fcpu = 16 Mhz**.

Q 2. Écrire la fonction `void transmet_mot(byte)` qui déclenche la transmission du mot fourni en paramètre. *(4 points)*

Voici les actions qu'elle réalise, dans l'ordre :

1. **Tester** l'autorisation de l'interruption : si elle est autorisée, c'est qu'une transmission est en cours. Il faut alors **attendre** qu'elle se termine, avant de poursuivre (« blocage » du programme = ok).
2. **Stocker** le mot à transmettre dans la variable globale `byte MOT` (on la suppose déjà déclarée ; pour rappel : `byte` = type entier 8 bits).
3. **Initialiser à 0** l'indice du bit à transmettre, dans la variable globale `int NUM_BIT` (déjà déclarée).
4. **Autoriser** l'interruption du timer 0 (celle dont la demande s'effectue 16 000 fois par sec.)

Q 3. Écrire la fonction d'interruption du timer 0, qui réalise la transmission d'un bit : **ISR(...)**. *(3 points)*

Indications :

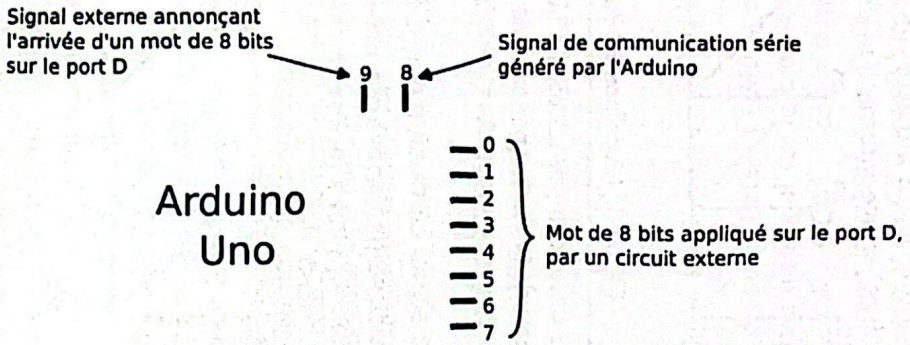
- La transmission du bit se fait sur la **broche 8 de l'Arduino Uno** (à l'aide d'une instruction).
- La fonction doit gérer l'évolution de la variable `NUM_BIT`, et transmettre le bit qu'elle désigne.
- La variable `NUM_BIT` - qui désigne le bit à transmettre - prend ses valeurs entre 0 et 10, car outre les 8 bits de `MOT`, il y a 3 bits de plus à transmettre (cf. protocole asynchrone).
- Si le bit envoyé était le **bit stop**, alors la transmission du mot est finie, et la fonction doit faire en sorte de **ne pas être rappelée**.

Partie 2 - Communication sérielle d'un mot reçu sur un port parallèle

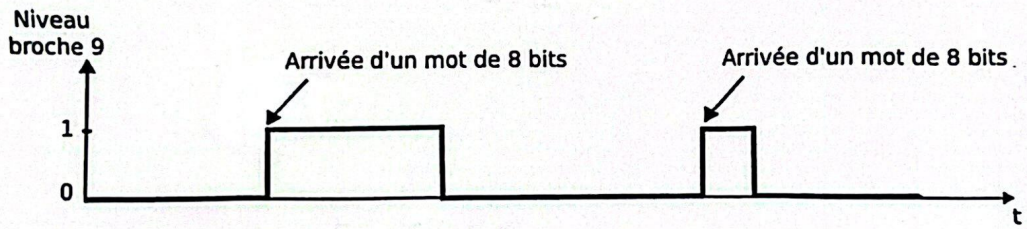
On va mettre en application la transmission de mot 8 bits réalisée dans la 1ère partie, pour transmettre une donnée reçue sur le port parallèle D (broches 0 à 7 de l'Arduino).

La réception d'un mot sur le port D (par communication parallèle), sera signalée par une **mise à l'état haut de la broche 9** (à l'état bas sinon).

Le schéma de branchement est représenté ci-dessous :



Et voici une représentation d'un signal reçu sur la broche 9, indiquant l'arrivée de 2 mots successifs sur le port parallèle :



On considère que tous les signaux en entrée du microcontrôleur ont des potentiels parfaitement définis (état haut + état bas) ; en effet, on suppose que ces signaux proviennent des sorties d'un autre microcontrôleur.

Q 4. Écrire la fonction `void init_ports`, qui configure les E/S du microcontrôleur *(3 points)*

Pour rappel, vous aurez besoin des broches arduino 0 à 7 (réception d'un mot par communication parallèle), de la broche 8 (émission des bits en série), et de la broche 9 (à scruter, pour savoir quand un mot est lisible sur le sur le port D).

Q 5. Écrire la fonction **void setup**, avec le strict minimum (aucune instruction superflue).*(1 point)*

Q 6. Écrire la fonction **void loop**, pour finaliser l'application. *(3 points)*

Dès qu'un mot est lisible sur le port D (évènement signalé sur la broche 9), alors il faut aussitôt transmettre ce mot sur la broche 8.

Faites attention à ne pas ré-émettre le même mot : il faut attendre le prochain front montant de la broche 9 avant de réitérer l'opération.