INSTITUT LIMAYRAC Toulouse

Synthèse sur la programmation des ports 1/2
Durée : 1h50

Objectifs	- Mettre en œuvre dans un même projet les principaux périphériques d'un microcontrôleur. - Mettre en œuvre la liaison série sous sa forme la plus simple : protocole de niveau liaison, codage en bande de base, transport avec des niveaux TTL. - Mettre en œuvre le port parallèle. - Mettre en œuvre le Timer.
Prérequis	- Le cours ATmega chapitre USART, Ports, Timers - Video d'introduction
Ressources à disposition	- PC station de travail sous Windows 10 - Carte Arduino Uno équipée d'un microcontrôleur ATmega328 - Document Le microcontrôleur ATMega328.pptx - Document Memento Représentations.pdf - Document Memento ATmega328.pdf - Barre 6 LEDs
Capacité / compétences visées	- S5 Solutions constructives des systèmes d'information - S5.1 Circuits Programmables complexes - S7 Réseaux, télécommunications et modes de transmission - S7.3 Protocoles de bas niveau, Synchrone/asynchrone, half/full duplex, bipoint/multipoints, …

1 Mise en situation

Dans les années 80, les premiers développeurs disposaient de ressources très limitées pour programmer des jeux. Typiquement, ils pouvaient programmer avec moins de 2Ko de mémoire. Ils y parvenaient en travaillant au plus proche du matériel, en manipulant directement les registres des périphériques.

Aujourd'hui, les objets connectés que nous pouvons imaginer autour des plus petits microcontrôleurs (ATmega328) disposent de la même puissance de calcul que les micro-ordinateurs de l'époque … et des mêmes limitations !

Dans les activités précédentes vous avez mis en œuvre des composants périphériques (Ports, Timers, Liaison série) en manipulant directement leurs registres. Dans cette activité vous allez programmer à la fois l'USART, le Timer, et le port parallèle dans l'objectif de réaliser un jeu simple.

Cette séance focalise sur l'USART et le port parallèle. Vous introduirez la gestion du timer dans la séance suivant afin d'estimer le niveau de rapidité du joueur.

2 Travail à faire

2.1 Récupération du programme

Clonez la librairie https://github.com/DaKprofSNir/BinMind en utilisant Github Desktop vers votre espace personnel.
Lancez l'environnement Arduino en ouvrant l'un des fichiers d'extention '.ino' du programme BindMind1.
L'onglet 'montage' explique comment connecter la barre LEDs, connectez-là.

Ce programme est encore incomplet, vous allez calculer les valeurs à placer dans les différents registres.
Une version finale a été téléversée dans vos Arduino afin que vous puissiez testez le programme final.

2.2 Objectif du programme

Vous allez écrire et exécuter un programme C qui :

- Tire un nombre au hasard entre zéro et 15 (0x0 et 0xF en hexadécimal),
- Ecrit ce nombre sur un port parallèle pour affichage sur une barre de LEDs,
- reçois la réponse du joueur par la broche RX du port série,
- fait echo de cette réponse par la broche TX du port série,
- réalise une animation fonction de l'exactitude ou non de la réponse du joueur.

en utilisant l'USART0 (Universal Synchronous/Asynchronous Receiver Transmitter n° zéro) du microcontrôleur ATmega328,
en utilisant le port B de l'ATmega 328.

2.3 Programmation de l'USART

Les paramètres de transmission sont :
9600 Baud, 8 bits de données, 1 bit de start, 2 bits de stop, pas de parité.
La vitesse se paramètre dans les registres UBRR0L et UBRR0H, le format dans le registre UCSR0C.

2.3.1 Paramétrage du format de transmission

Le registre UCSR0C contient 8 bits. Référez-vous au mémento ATmega afin de déterminez leur valeur et complétez le tableau :

UCSR0C			
N° bit	Nom	Valeur du bit	Rôle du bit
0	UCPOL0	0	front à choisir pour échantillonner
1	UCSZ01		Taille de la donnée (de 5 à 8 bits)
2	UCSZ02		Taille de la donnée (de 5 à 8 bits)
3	UCSBS0		Nombre de bits de stop
4	UPM00		Parité paire, impaire, ou sans
5	UPM01		Parité paire, impaire, ou sans
6	UMSEL00	0	Mode asynchrone
7	UMSEL01	0	Mode asynchrone

Quelle est la valeur à placer dans UCSR0C en hexadécimal ? .....

2.3.2 Paramétrage de la vitesse

Le registre initialisant comporte deux octets, UBRR0L et UBRR0H. Pour calculer la vitesse on utilise l'équation UBRR0 = (SysClock / (16 * bauds)) -1

Si le résultat est assez petit pour tenir dans un octet, il viendra dans UBRR0L et le poids fort UBRR0H sera à zéro.

Quelle est la valeur à placer dans UBRR0L en hexadécimal ? .....

Quelle est la valeur à placer dans UBRR0H en hexadécimal ? .....

2.4 Initialisation du port B

Lorsque la barre LED est montée sur les broches 8 à GND de la carte Arduino, chaque LED se trouve connectée à un bit du port B.

Les registres du port B comportent 8 bits (comme tous les registres) mais seuls les 6 bits b0 à b5 sont connectés vers l'extérieur du microcontrôleur (il aurait fallu deux broches de plus mais le constructeur a préféré rester dans un format de composant standard à 28 broches).

Nous avons besoin de 4 bits pour afficher le nombre à deviner, mais nous utiliserons les 6 bits pour commander les 6 LEDs dans nos animations.

Vous devez placer les 6 bits du port B en sortie, les bits inutilisés devant toujours, par sécurité, être programmés en entrée. Déterminer les valeurs à placer dans le registre de direction du port B en complétant le tableau, sachant que 0 signifie entrée, et 1 signifie sortie.

DDRB	b7	b6	b5	b4	b3	b2	b1	b0
Valeur								

Quelle valeur faudrait-il placer dans le port pour allumer une LED sur 2 ?

PORTB	b7	b6	b5	b4	b3	b2	b1	b0
Valeur								
Soit en hexadécimal :								

2.5 Paramétrage des périphériques dans le programme

Reportez les valeurs calculées et téléversez le programme.

Testez votre programme. Il doit se comporter comme attendu.

2.6 Compréhension du programme.

Notez que la boucle while tourne tant que le joueur n'a pas donné la bonne réponse.

2.6.1 Programmation du port parallèle

Ces deux lignes de codes font afficher le nombre en allumant complétement les 1 et faiblement les 0

```
//--- Afficher le nombre à convertir en hexa ----
PORTB = 0x0F;   delay( 1); // les 0 vont briller léger
PORTB = nombre; delay(24); // les 1 vont briller au max
```

Quelle instruction allume les 4 LEDs ?.....

Quelle instruction envoie l'octet *nombre* vers le port B ? .....

L'instruction delay() effectue une attente qui dure n millisecondes.

Expliquez pourquoi les bits à zéro dans *nombre* sont plus faiblement allumés que les bits à 1 :

.....

.....

D'autres exemples de manipulation du port parallèle, utilisant des opérateurs binaires, sont visible dans l'onglet *animation*. Vous pouvez vous y référer par curiosité.

2.6.2 Programmation de l'USART

Dans le fragment de code suivant on teste un bit afin de savoir si le joueur a envoyer une réponse

```
//---- lire la réponse du joueur dans l'USART
if( (UCSR0A & 0x80) != 0 ){ // on a reçu un caractère
    reponse = UDR0; // lecture du registre d'entrée de l'USART
```

À quoi sert le registre UCSR0A ? .....

Quel est le nom du bit testé ? .....

Quel est le rôle de ce bit ? .....

Quelle est la valeur de *(UCSR0A & 0x80)* lorsqu'un caractère a été reçu dans UDR0 ? .....

Quelle est la valeur de *(UCSR0A & 0x80)* lorsque UDR0 est encore vide ? .....

À quel moment le bit _____ est-il remis à zéro ? .....

3 Conclusion et bilan

Que pensez-vous avoir appris ou approfondis plus particulièrement dans cette activité ?

.....

.....

.....

.....

.....

4 Questions facultatives

On désire mesurer le temps pris par le joueur pour répondre juste, afin de lui afficher son "niveau".
Imaginez un algorithme utilisant un timer pour effectuer cette mesure,
Imaginez un moyen d'afficher ce niveau en utilisant la barre LED.

.....

.....

.....

.....

.....

.....