TP 03 Timers and interrupts

Equipe NIST : Rayane Bait, Guilhem Mizrahi

Partie 1

Question 1

Macros préprocesseur :

- DDRB -> (*(volatile uint8_t *)((0x04) + 0x20))
- _BV -> (DDB5) (1 << (5))

Sections du ELF :

[Nr]	Name	Type	Addr	Off	Size	ES	Flg	Lk	Inf	Al
[0]		NULL	0	0	0	0		0	0	0
[1]	.data	PROGBITS	800100	124	0	0	WA	0	0	1
[2]	.text	PROGBITS	0	74	0000b0	0	AX	0	0	2
[3]	.comment	PROGBITS	0	124	11	1	MS	0	0	1
[4]	.note.gnu.avr.dev	NOTE	0	138	40	0		0	0	4
[5]	.debug_info	PROGBITS	0	178	0005f4	0		0	0	1
[6]	.debug_abbrev	PROGBITS	0	00076c	0005a2	0		0	0	1
[7]	.debug_line	PROGBITS	0	000d0e	00001a	0		0	0	1
[8]	.debug_str	PROGBITS	0	000d28	208	0		0	0	1
[9]	.shstrtab	STRTAB	0	0016ed	00007a	0		0	0	1
[10]	.symtab	SYMTAB	0	000f30	480	10		11	17	4
[11]	.strtab	STRTAB	0	0013b0	00033d	0		0	0	1

La section .text contient le programme à exécuter.
La section .data est vide mais elle pourrait contenir des données à initialiser avant d'entrer dans le programme.

Séquence d'entrée dans le programme :

Avant d'entrer dans la fonction main il y a l'initialisation des interruptions via trois parties de code :

1. __vectors
2. __ctors_end
3. __bad_interrupt

- La première définit vers quoi pointent les différentes interruptions. Seule la première est réellement définie, les autres pointent sur la première.
- La deuxième section désactive les interruptions en mettant 0 dans SREG (0x3F) et initialise la stack à 0x8FF (2kB). Ensuite, la fonction main est appelée.
- La troisième section permet de faire pointer les autres interruptions sur la première.

Question 2

Le code source du programme se trouve dans le fichier `exercice1.c` et se compile en utilisant le Makefile de la façon suivante : `make PROG=exercice1`

Partie 2

Question 1 :

D'après la section 7.3.1 de la documentation générale du microprocesseur (page 20), le bit de Global Interrupt Enable doit être activé dans le registre SREG. D'après la section 5.104 du set d'instructions, l'instruction SEI active ce bit. En AVR-libc, c'est la macro `sei()`.

Question 2 :

En mode normal, le conteur commence a 0 et le flag TOV est activé au moment où le compteur repasse à 0, ce qui prend 256 cycles car le registre fait 8 bits. a la fréquence de 16 MHz, l'intervalle de temps est de 16 microsecondes.

Question 3 :

Les réglages de prescaler sont indiqués dans la section 17.7. Au maximum la fréquence peut être divisée par 1024. Avec une telle division de fréquence, l'intervalle de temps entre deux interruptions est de 16.384 millisecondes.

Question 4 :

En mode normal, a 16 MHz, l'intervalle de temps entre deux interruptions est de 65536 / 16 000 000 = 4.096 millisecondes. Avec le prescaler réglé à 1024, l'intervalle de temps est de 4.194304 secondes.

Question 5 :

Avec le prescaler réglé à 64, le temps maximum entre deux interruptions est de 262.144 millisecondes. Il faut donc régler le prescaler à 256. Avec un tel réglage, mettre la valeur 31250 dans OCR1A permet de déclencher une interruption toutes les 500 millisecondes.

Question 6 :

Pour permettre d'attendre des heures ou des jours, il faut implémenter une boucle qui compte le nombre d'intervalles de 4 secondes. Par exemple, pour attendre une heure, avec le prescaler à 1024 et OCR1A à 62500, une interruption se produit toutes les 4 secondes. En comptant 900 interruptions, on peut compter un délai d'une heure.

Partie 3

Le code source du programme se trouve dans le fichier `exercice2.c` et se compile en utilisant le Makefile de la façon suivante : `make PROG=exercice2`

Avantages :

L'avantage d'utiliser un timer par rapport à une attente active est de pouvoir mettre le microprocesseur en mode IDLE et de consommer moins d'énergie. Ainsi, en embarqué, la durée d'autonomie électrique du système est plus longue. D'autre part, le Timer permet une assez grande flexibilité dans le choix du délai comparée au Watchdog Timer.

Désavantages :

L'utilisation du Timer ne permet pas de désactiver tous les modules et est incompatible avec un sommeil trop profond.

Partie 4

Le code source du programme se trouve dans le fichier `exercice3.c` et se compile en utilisant le Makefile de la façon suivante : `make PROG=exercice3`

Avantages :

L'avantage d'utiliser le Watchdog Timer par rapport a un timer est de pouvoir désactiver plus de fonctionnalités et de rentrer dans un mode de sommeil plus profond, permettant de consommer moins d'énergie.

Désavantages :

Cependant, le WDT ne permet pas une grande finesse dans le choix du délai (10 valeurs possibles entre 16 ms et 8s) et ne garantit pas une grande précision du délai qui peut dépendre de l'alimentation du système et de la température (cf. Fig 33-34 page 566).