

MICROCONTRÔLEURS MICROCONTRÔLEURS ET SYSTÈMES NUMÉRIQUES TRAVAIL PRATIQUE NO 6

MT GROUPE A	MT Groupe B	EL		
No du Groupe	Premier Etudiant	Second Etudiant	Evaluation	Visa Correcteur
093	Nathann Morand	Felipe Ramirez		

6. INTERRUPTIONS

Les interruptions sont le mécanisme dont dispose un microcontrôleur lui permettant de réagir rapidement à des événements externes ou internes. Le but de ce travail pratique est d'acquérir la connaissance sur les différents interruptions disponibles sur ATmega128, leur programmation, et leur déroulement.

6.1 LE VECTEUR D'INTERRUPTION, PRIORITÉ DES INTERRUPTIONS

Les quatre lignes d'interruptions externes INT0...INT3 (barre car actif bas) se trouvent sur les pins 0 à 3 du port D. Les interruptions INT4...INT7 se trouvent sur les pins 4 à 7 du port E. Ces dernières sont associées à un registre de contrôle nommé EICR qui permet d'en programmer la sensibilité.

Deux étapes sont nécessaires à l'autorisation d'une interruption:

- le bit correspondant à l'instruction doit être activé dans le registre nommé EIMSK, signifiant External Interrupt Mask register, et qui se trouve à l'adresse \$39;
- les interruptions doivent être globalement autorisées en mettant le i dans SREG à 1, par exemple au moyen de l'instruction sei signifiant SET global Interrupt flag.

Si les interruptions INT0...INT3 sont autorisées, alors un niveau logique ‘0’ présenté sur une de ces lignes force le microcontrôleur à suspendre le programme principal courant et à sauter à une routine spéciale de service de l'interruption (en anglais ISR signifiant Interrupt Service Routine).

Le microcontrôleur répond à une interruption de façon autonome par l'exécution des tâches suivantes, chronologiquement:

- l'exécution de l'instruction courante se termine;
- les interruptions sont désactivées et le PC est sauvegardé sur la pile puis le programme saute à l'adresse du vecteur d'interruption;
- grâce à l'ISR on jump à l'adresse de la routine concernée par l'interruption détectée;
- exécution de la routine;
- lorsque que ret est détecté, l'interruption est réactivée et le PC est récupéré de la pile.

Chargez le fichier int0c.asm en Figure 6.1 dans AtmelStudio, compilez-le et téléchargez-le.

```
; file      int0c.asm    target ATmega128L-4MHz-STK300
; purpose using INT0..INT3
.include "macros.asm"          ; include macros definitions
.include "definitions.asm"     ; include register/constant definition

; === interrupt table ===
.org 0
    jmp  reset
.org INT0addr
    jmp  ext_int0
.org INT1addr
    jmp  ext_int1
.org INT2addr
    jmp  ext_int2
.org INT3addr
    jmp  ext_int3

; === interrupt service routines
ext_int0:
    cbi  PORTB,0  ; turn on LED 0
    reti
ext_int1:
    cbi  PORTB,1  ; turn on LED 1
    reti
ext_int2:
    cbi  PORTB,2  ; turn on LED 2
    reti
ext_int3:
    cbi  PORTB,3  ; turn on LED 3
    reti

; === initialization (reset) ====
reset:
    LDSP    RAMEND           ; load stack pointer SP
    OUTI    DDRB, 0xFF        ; portB = output
    OUTI    EIMSK, 0b00001111 ; enable INT0..INT3;
    sei                 ; set global interrupt

; === main program ===
main:
    WAIT_US  10000           ; wait 10 msec
    dec     r18               ; decrement counter
    out    PORTB,r18         ; output counter value to LED
    rjmp   main
```

Figure 6.1: int0c.asm

Les fusibles doivent être programmé en suivant rigoureusement la procédure présentée en Figure 2.6 (TP02).

Sans appuyer sur les boutons PD0...PD3 le programme exécute uniquement le partie principale avec l'étiquette [main]. Lorsqu'un des interrupteurs est activé avec persistance, le programme principal est constamment interrompu et doit exécuter la routine d'interruption correspondante de façon répétée.

Que se passe-t'il si plusieurs interruptions sont actives simultanément; y-a-t'il une forme de priorité?

Testez-le au moyen de la carte STK300 !

les interruption se font par ordre suivant : PD0 est la plus prioritaire suivis de PD1 et PD2 finalement, PD3 est la dernière

6.2 TABLE DES VECTEURS D'INTERRUPTIONS

A quoi servent les lignes de code .org INTXaddr dans int0c.asm ?

elle attache l'adresse de la routine de service à l'adresse de la table du vecteur d'interruption"

Mettez les instructions de saut des interruptions (INT5, TIM2_OVF, et USART1_RXC) à la bonne adresse dans la table des vecteurs d'interruption, au début du programme en Figure 6.2. Ecrivez les adresses en hexadécimal. .

```
; === interrupt table ===
.org 0x0000
    jmp reset

.org 0x000c
    jmp ext_int5

.org 0x0014
    jmp tim2_ovf

.org 0x003c
    jmp USART1_RXC
```

Figure 6.2: Portion de la table d'interruption du ATmega128.

Dans quel document peut-on trouver cet information ?

m128dec.inc

Ces adresses correspondent'elles à une implémentation matérielle ou logicielle ?

les addresse correspondent a une implementation matérielle

A quelle adresse se trouve la routine de service de l'interruption externe INT5, ou exprimé de façon différente, à quelle adresse le programme branche-t'il à l'exécution du jmp ext_int5 ?

à l'adresse de la routine ext_int5. dans la mémoire programme.

6.3 AUTORISATION DES INTERRUPTIONS

Ecrivez le morceau de code qui autorise les interruptions INT0, INT2, INT5, INT6 à l'exclusion des autres.

```
ldi r16, 0b01100101
out EIMSK, r16
```

Ecrivez le morceau de code qui autorise les interruptions INT0, INT1, INT5, INT7 sans modifier l'état des autres.

```
in r16, EIMSK
ori r16, 0b10100011
out EIMSK, r16
```

Ecrivez le morceau de code qui bloque les interruptions INT1, INT3 sans modifier l'état des autres.

```
in      r16, EIMSK
[andi]  r16, ~$0a
out    EIMSK, r16
```

6.4 CONDITIONS DES INTERRUPTIONS

Ecrivez le bout de code qui autorise les interruptions INT5, INT6, INT7, et qui configure INT5 pour un flanc montant, INT6 pour un flanc descendant et INT7 pour un niveau ‘low’.

```
ldi      r16, 0b0b11100000
out    [EIMSK], r16 ; external interrupt mask
ldi      r16, 0b000101100
out    [EICR], r16 ; external interrupt control register
```

Ecrivez le bout de code qui autorise les interruptions INT4, INT5, et qui configure INT4 pour un flanc montant, INT5 pour un flanc descendant et qui laisse les autres interruptions déjà configurées dans leur état.

```
in      r16, EIMSK
ori    r16, 0b00110000
out    [EIMSK], r16 ; set external interrupt mask
in      r16, EICRB
[andi] r16, 0b1111011 ; set the 0s
[ori]  r16, 0b00001011 ; set the 1s
out    [EICR], r16; external interrupt control register
```

6.5 SIMULATION DES INTERRUPTIONS EXTERNE

Il est possible de simuler les interruptions externes et en observer les effets sur le déroulement du programme. Chargez le programme int0b.asm en Figure 6.4 et simulez-le (ne le téléchargez pas !). AtmelStudio 7 doit être configuré pour accepter les interruption lorsque la simulation est effectuée en mode pas-à-pas. En activant Tools→Options, une fenêtre de configuration apparaît (Figure 6.3); le fanion Mask interrupts while stepping est par défaut en True, et doit être changé en False.

Utilisez la fenêtre qui vous permet de visualiser les I/Os afin d'observer l'état des port, et de les modifier pour simuler des requêtes d'interruptions.

Vérifiez les points suivants, en simulation pas à pas:

- 1) Vérifiez que le programme n'effectue qu'une seule instruction du programme principal, puis effectue la procédure d'interruption. Quelle est la raison de ce comportement ? Quelle(s) interruption(s) est(sont) active(s) ?

- 2) Mettez les bits 0 à 2 de PIND au niveau logique ‘0’ et confirmez l'ordre de priorité des interruptions.
- 3) Mettez le bit I dans SREG au niveau logique ‘0’ afin de bloquer toutes les interruptions. Que constatez-vous ?

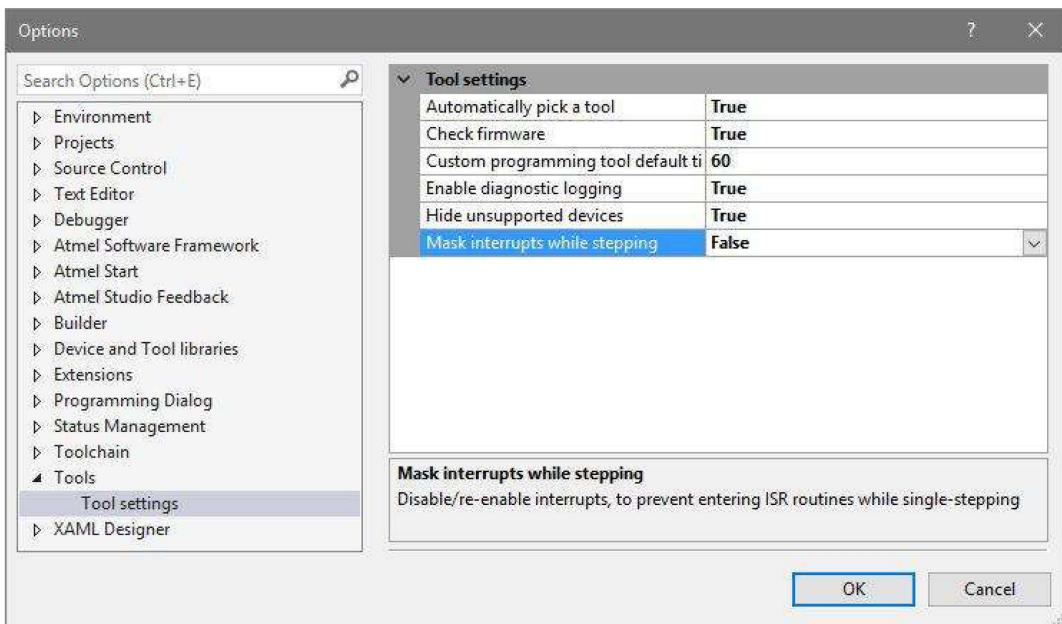


Figure 6.3: Configuration de AtmelStudio 7.

- 4) Mettez au niveau logique ‘0’ les bits 0 à 3 de PIND. Mettez le bit de contrôle INT0 dans EIMSK au niveau logique ‘0’ et vérifiez que l’interruption 0 soit bloquée. Quelle interruption est alors servie ?
INT1
- 5) Mettez PIND0 à PIND3 successivement au niveau logique ‘1’ et observez le changement de comportement dans le déroulement du programme.
- 6) L’interruption INT4 est déclenchée par une transition du niveau logique ‘1’ à ‘0’ sur la ligne PE4. Vérifiez ce comportement en simulant en répétant le pas de simulation (F11), et changeant les valeurs sur PE4 (après avoir désactivé les interruptions sur PIND).
- 7) L’interruption INT5 est déclenchée par une transition du niveau logique ‘0’ à ‘1’ sur la ligne PE5. Vérifiez ce comportement, de même.

```

; fileint0b.asm      target ATmega128L-4MHz-STK300
; purpose simulation of external interrupts

.include "macros.asm"          ; include macros definitions
.include "definitions.asm"     ; include register definitions

; === interrupt table ===
    jmp  reset
    jmp  int_0      ; PIND0..3
    jmp  int_1
    jmp  int_2
    jmp  int_3
    jmp  int_4      ; PINE4..7
    jmp  int_5
    jmp  int_6
    jmp  int_7

; === interrupt service routines
int_0:   inc  r0
         reti
int_1:   reti
int_2:   reti
int_3:   reti
int_4:   inc  r1
         reti
int_5:   reti
int_6:   reti
int_7:   reti

; === initialization (reset) ====
reset:LDSRAMEND; load SP
    OUTI EIMSK, 0b00111111 ; INT0..5
    OUTIEICRA, 0b00000000 ; INT3..0; 00=low, 10=fall, 11=rise
    OUTI EICRB, 0b00001110 ; INT7..4
    sei                  ; set global interrupt

; === main program ===
main:  inc r16
       nop
       rjmp main

```

Figure 6.4: int0b.asm.

6.6 OBERVATION DES DIAGRAMMES DES TEMPS (TIMINGS) À L'OSCILLOSCOPE

Il est possible d'observer les différents délais associés à l'exécution d'une interruption, ainsi que le déroulement de l'exécution au moyen de l'oscilloscope. Le programme int1.asm donné en Figure 6.5 permet cette observation car il adresse un port en sortie dans le programme principal, et un autre port en sortie dans la routine de service de l'interruption. Il suffit alors d'observer l'état des ports, et d'associer les timings obtenus aux instructions exécutées. C'est le but de l'exercice proposé.

```

; file int1b.asm    target ATmega128L-4MHz-STK300
; purpose oscilloscope measurements while externally interrupting
.include "macros.asm"           ; include macros definitions
.include "definitions.asm"       ; include register definitions

; === interrupt table ===
.org 0
    jmp    reset

.org INT3addr
    jmp    ext_int3

; === interrupt service routines
ext_int3:
    sbi    PORTA,5          ; pulse on PINA5
    nop
    cbi    PORTA,5
    reti

; === initialization (reset) ====
reset:
    LDSP   RAMEND          ; load stack pointer SP
    OUTI   DDRA, 0b00100010 ; PA1,PA5 = output
    OUTI   EIMSK,0b00001000 ; enable INT3;
    sei               ; set global interrupt

; === main program ===
main:
    sbi    PORTA,1          ; pulse on PINA1
    nop
    cbi    PORTA,1
    rjmp   main

```

Figure 6.5: int1b.asm.

6.6.1 PRÉPARATION DU MONTAGE

- 1) Connectez la sonde 1 de l'oscilloscope à la pin PA1. Dans quel document trouvez-vous le plan de connection ?

dans la schematic de la carte STK
- 2) Connectez la sonde 2 de l'oscilloscope à la pin PA5.
- 3) Placez le diviseur sur chaque sonde sur la position 10X.
- 4) Configurez l'oscilloscope avec les valeurs données au bas de la copie d'écran en Figure 6.6 et Figure 6.7, soit [5.00V/div, 500ns/div] et [5.00V/div, 1us/div] respectivement.
- 5) Réglez la position verticale (VERTICAL POSITION) des la trace comme indiqué sur les Figure 6.6 et Figure 6.7. Ainsi, le niveau de la masse est donné par les positions respectives des flèches pour chaque canal: 1→, 2→.
- 6) Configurez les paramètres d'aquisition des canaux dans les menus CH1 et CH2 comme suit:
 - sélectionnez Coupling DC;
 - sélectionnez BW(Bandwidth) Limit OFF;

- sélectionnez Volts/Div Coarse;
 - sélectionnez Probe 10X;
 - sélectionnez Invert OFF.
- 7) Configurez les paramètres du trigger dans le menu TRIGGER comme suit:
- sélectionnez Edge (trigger sur des flancs);
 - sélectionnez Rising (trigger sur un flanc montant);
 - sélectionnez Source CH1 (trigger sur le canal 1, pin PA1 contrôlée par le programme principal);
 - sélectionnez Mode Normal et placez le niveau du trigger TRIGGER LEVEL sur 2.5V (au milieu de la gamme, donné par la flèche ←);
 - sélectionnez Coupling DC.

6.6.2 MANIPULATIONS

Téléchargez int1b.asm sur le microcontrôleur. Observez les signaux générés en l'absence d'interruption.

Reportez sur la Figure 6.6 les deux traces que vous observez à l'oscilloscope. Associez aux traces visibles les instructions suivantes du programme principal (main):

- sbi PORTA,1
- nop
- cbi PORTA,1
- rjmp main

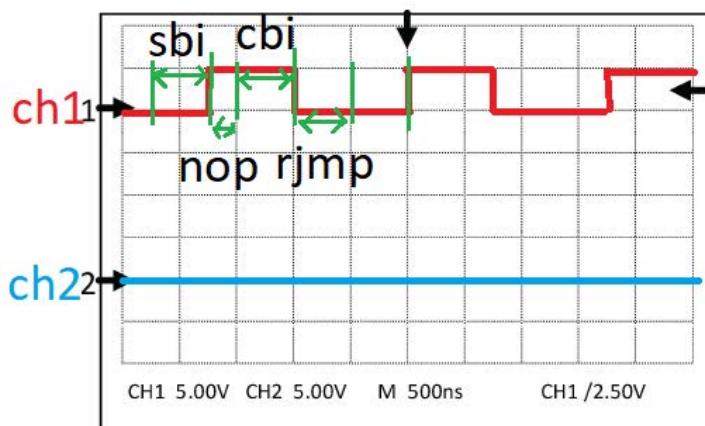


Figure 6.6: Traces visibles à l'oscilloscope: sans interruption active.

Appuyez sur le bouton PD3 avec persistance afin d'activer INT3. L'exécution du programme alterne alors entre le programme principal, et la routine de service de l'interruption. Considérez le code software qui contrôle la génération de signaux, la trace 1, correspondant à la pin PA1 est contrôlée par **le main, la boucle principale** alors que la trace 2, correspondant à la pin PA5 est contrôlée par **la sous routine d'interruption**.

Modifiez la base de temps à 1.0us, provoquez l'interruption et maintenez-la, puis au moyen de la mollette HORIZONTAL POSITION, déplacez la trace 1 de façon à centrer le pulse obtenu comme indiqué en Figure 6.7. Reportez la trace 2. Associez les instructions suivantes au diagramme temporel obtenu, et indiquez à chaque fois le nombre de cycles, en vérifiant bien d'avoir le total exact de cycles nécessaires correspondant à la trace visible:

- sbi PORTA,1
- saut au vecteur d'interruption
- jmp ext_int3

- sbi PORTA,5
- nop
- cbi PORTA,5
- reti
- nop
- ...
- cbi PORTA,1

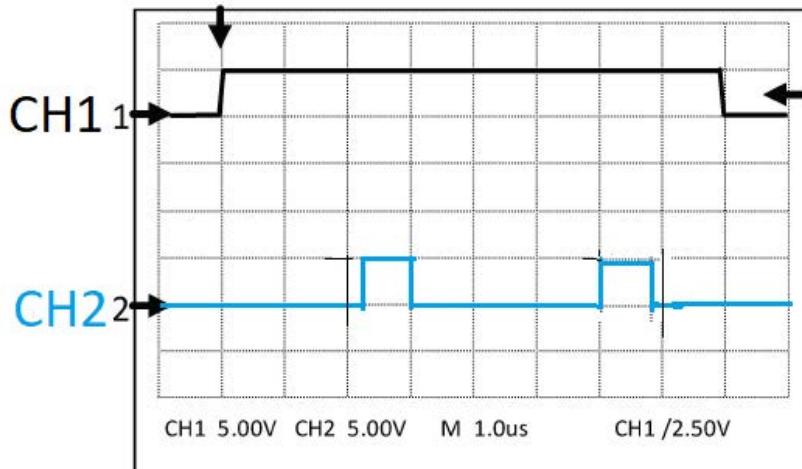


Figure 6.7: Traces visibles à l'oscilloscope: avec interruption active.

Dans le cas où il n'y a pas d'interruption, l'impulsion au niveau haut sur la trace 1 dure cycle(s); alors que dans le cas où il y a une interruption (maintenue), l'impulsion sur la trace 1 dure cycles.

On observe plusieurs impulsions de la trace 2 qui ont lieu pendant une impulsion unique de la trace 1. Expliquez et justifiez le nombre d'impulsions observées sur la trace 2 !

il y a 2 interruption. la première apres sbi PORTA 1 et la seconde apres le nop (les interruptions se répète en boucle tant que la pin reste l'état 0)

