Laboratoire 2 : Interruptions matérielles. Interruptions externes

Ce laboratoire est destiné à vous familiariser avec le travail avec les interruptions matérielles et en particulier avec les interruptions externes. Nous utiliserons des interruptions externes pour détecter immédiatement (en temps réel) une pression sur un bouton, indépendamment du programme principal.

1. Interruptions

Une interruption matérielle est un signal synchrone ou asynchrone d'un périphérique qui signale l'occurrence d'un événement qui doit être géré par le processeur. La gestion des interruptions a pour effet de suspendre le fil d'exécution normal d'un programme et de lancer une **routine de gestion des interruptions (RTI)** .

Des interruptions matérielles ont été introduites pour éliminer les boucles qu'un processeur devrait effectuer en attendant un événement d'un périphérique. Grâce à un système d'interruption, les périphériques peuvent communiquer avec le processeur, laissant le processeur libre d'exécuter son programme normal le reste du temps et n'interrompant son exécution que lorsque cela est nécessaire.

Avant d'exécuter un RTI, le processeur doit disposer d'un mécanisme pour enregistrer l'état dans lequel il se trouvait lorsque l'interruption s'est produite. Cela se fait en sauvegardant dans une mémoire, le plus souvent organisée sous forme de pile, le registre du compteur de programme (Program Counter), les registres d'état ainsi que toutes les variables du programme qui sont affectées par l'exécution du RTI. A la fin de l'exécution du RTI, l'état précédent des registres est restauré et le programme principal reprend à partir du point où il a été interrompu.

Pour associer une interruption à une routine spécifique du programme, le processeur utilise **la table des vecteurs d'interruption** (TVI), illustrée dans la figure ci-dessous. Dans cette table, chaque interruption est associée à l'adresse de son gestionnaire auquel le programme sautera lorsqu'elle se produira. Ces adresses sont prédéfinies et sont mappées dans la mémoire programme dans un espace contigu qui constitue le TVI. Les adresses d'interruption dans TVI sont définies en fonction de leur priorité : plus l'adresse est faible, plus la priorité est élevée.

Vecteur non.	Adresse du programme	Source	Définition d'interruption
1	0000	RÉINITIALISER	Broche externe, réinitialisation à la mise sous tension, réinitialisation de la coupure de courant, réinitialisation du chien de garde et réinitialisation JTAG AWR
2	0002	INT0	Demande d'interruption externe 0
3	0004	INT1	Demande d'interruption externe 1
4	0006	PCINT0	Demande d'interruption de changement de broche 0
5	0008	PCINT1	Demande d'interruption de changement de broche 1
6	000A	PCINT2	Demande d'interruption de changement de broche 2
7	000C	WDT	Interruption de temporisation du chien de garde
8	000E	MINUTERIE2_COMPA	Minuterie/Compteur2 Comparer Match A
9	0010	MINUTERIE2_COMPB	Minuterie/Compteur2 Comparer Match B
dix	0012	MINUTERIE2_OVF	Temporisateur/Compteur 2 Débordement
11	0014	MINUTERIE1_CAPT	Événement de capture du minuteur/compteur1
12	0016	MINUTERIE1_COMPA	Minuterie/Compteur1 Comparer Match A
13	0018	MINUTERIE1_COMPB	Minuterie/Compteur1 Comparer Match B
14	001A	MINUTERIE1_OVF	Temporisateur/Compteur1 Débordement
15	001C	MINUTERIE0_COMPA	Minuterie/Compteur0 Comparer Match A
16	001E	MINUTERIE0_COMPB	Minuterie/Compteur0 Comparer Match B
17	0020	MINUTERIE0_OVF	Temporisateur/Compteur0 Débordement
18	0022	SPI_STC	Transfert série SPI terminé
19	0024	USART0_RX	USART0 Rx terminé
20	0026	USART0_OUTER	Registre de données USART0 vide
21	0028	USART0_TX	USART0 Tx terminé
22	002A	ADC	Conversion ADC terminée
23	002C	EE_PRÊT	EEPROM prêt
24	002E	ANALOG_COMP	Comparateur analogique
25	0030	TWI	Interface série à deux fils
26	0032	SPM_READY	Mémoriser la mémoire de programme prête

Lorsqu'une interruption se produit, outre la sauvegarde de l'état, le processeur désactive les interruptions et, au retour de la routine de traitement, il les réactive. Leur activation peut aussi être forcée depuis le gestionnaire d'interruptions (par exemple, on est dans le gestionnaire de réception d'une trame sur l'interface série, et on veut activer les interruptions temporisées).

Utilisation des interruptions

En général, les étapes pour activer une interruption sont les suivantes :

Nous démarrons le mécanisme de gestion des interruptions

Le mécanisme d'interruption doit être activé explicitement (bit I - Global Interrupt Enable dans le registre SREG). Pour définir et réinitialiser ce bit, il existe également deux fonctions d'assistance :

```
// active les interruptions
sei ( ) ;
// désactiver les interruptions
cli ( ) ;
```

En général, les étapes pour activer une interruption INT ou PCINT externe sont les suivantes :

On configure le périphérique qui enverra les interruptions

Dans l'exemple, pour INT0et INT1(broches PD2et PD3), la configuration se fait à partir du registre EICRA(External Interrupt Control Register A, section 12.2.1, page 54 de la fiche technique), tandis que pour les interruptions de changement de broche, le registre est utilisé PCICR.

```
// interruptions externes
EICRA |=  ( 1  << ISC00 ) ;     // définit INT0 pour qu'il se déclenche sur TOUT changement logique
// interruptions de type changement de broche (vecteur d'activation des interruptions)
PCICR |=  ( 1  << PCIE2 ) ;  // activer l'interruption de changement de broche, définir PCIE2 pour activer le scan PCMSK2
// autres interruptions
```

Une interruption externe peut être configurée pour **détecter la transition du signal logique** sur un front montant, descendant ou l'un ou l'autre (voir tableau 13-1 dans la fiche technique). En comparaison, une interruption de changement de broche est générée lorsque **le niveau logique** du signal change, et la valeur du signal **doit être vérifiée explicitement dans le code d'interruption** .

Nous écrivons la routine de gestion des interruptions

Par exemple, pour les interruptions de type INT0et , respectivement PCINT2, le code ressemblerait à ceci :

```
ISR ( INT0_vect )
{
  // code d'interruption externe PD2 /INT0
  // vérification de la transition sur front montant, descendant ou sur n'importe quel front
  // (comme INT0 est configuré)
}

ISR ( PCINT2_vect ) {
  // changement de code d'interruption de type de broche
  si  ( ( PIN &  ( 1  << PD4 ) )  ==  0 ) {
     // l'interruption a été générée par la broche PD4 / PCINT20
     // vérifier le niveau logique de la broche
  }
  // l'interruption a été générée par une autre broche
}
```

Si nous avons plusieurs interruptions sur le même vecteur et par ex. plusieurs boutons, il est impossible de déterminer exactement quel bouton a généré l'interruption PCINT. De plus, il est possible que plusieurs interruptions se produisent alors que l'état d'une broche cochée reste inchangé, auquel cas une nouvelle poussée sera détectée par erreur.

Nous activons l'interruption et testons le programme

Dans le cas présent, pour activer les interruptions externes, configurez le registre de masque d'interruption externe (EIMSK) comme suit :

- les bits INT1:0contrôlent si les interruptions externes (INT0-INT1) sont activées

Pour les autres interruptions, configurez le registre approprié (par exemple PCMSK2pour activer les interruptions de changement de broche sur chaque broche du port D)

```
// interruption externe
EIMSK |=  ( 1  << INT0 ) ;      // Active INT0
// interruption de changement de broche
PCMSK2 |=  ( 1  << PCINT20 ) ;  // Active PCINT20 (PD4)
// active les interruptions globales
sei ( ) ;
```

Travailler avec des interruptions

Règles de programmation en contexte d'interruption :

- Il n'y a pas de valeur de retour. À la fin de l'exécution du gestionnaire, le processeur reprend l'exécution des instructions là où il s'était arrêté avant le déclenchement de l'interruption
- Étant donné que le gestionnaire retarde toute autre activité dans main et inhibe l'exécution d'autres interruptions, **le temps d'exécution doit être le plus court possible** .
- Lors de l'utilisation de variables communes dans plusieurs gestionnaires d'interruptions et/ou principaux, il faut envisager **un accès simultané** à celles-ci (en particulier aux variables 16/32 bits dont la modification nécessite plusieurs cycles processeur).
- Les variables partagées doivent être marquées afin `volatile`que leurs accès ne soient pas optimisés par le compilateur

Les autres wrappers (autour de certaines instructions de base AVR) fournis par l'interface sont :

- **sei()** - définit le bit I de SREG sur 1, activant les interruptions globales. Appelez l' instruction assembleur *sei*
- **cli()** - définit le bit I de SREG sur 0, désactivant les interruptions globales. Appelle l' instruction *cli de l'assembleur*
- **reti()** - revient d'un gestionnaire d'interruptions, active les interruptions globales. Il doit s'agir de la dernière instruction appelée dans une routine qui utilise l'indicateur ISR_NAKED.

Les interruptions rencontrées qui n'ont pas de routine de gestionnaire provoqueront une interruption de réinitialisation !

2. Interruptions externes. INT contre PCINT

Les interruptions peuvent être divisées en deux catégories :

- interruptions des composants *internes* : temporisateurs, interfaces série (USART), convertisseur analogique-numérique
- interruptions de composants *externes* : activées par des broches uC externes

Dans le Lab 2 , vous avez étudié le mécanisme d'interruption sur l'Atmega328p et configuré les interruptions dans le contexte des minuteries.

En plus des interruptions des composants internes de l'uC, il existe également des lignes pour les interruptions des périphériques externes : INT0-INT1 et PCINT0-PCINT2. La différence entre ces deux types d'interruptions externes est donnée par les capacités prises en charge et leur granularité.

Les signaux des interruptions **INTn** peuvent générer une interruption sur transition (montante, descendante ou les deux) ou au niveau 0, selon la configuration de l'interruption. **Les** interruptions PCINTn (Pin Change INTerrupt) sont déclenchées sur les deux transitions (en particulier, lorsque la valeur est basculée) et 8 broches sont multiplexées sur une seule interruption. Les signaux d'interruption PCINT peuvent être activés individuellement, mais pour savoir exactement quel signal a déclenché une interruption particulière, le registre PINn correspondant doit être vérifié. Si plusieurs signaux sont déclenchés en même temps, ils ne pourront pas être distingués.

Dans le cas d'interruptions de type interruption de changement de broche, ne confondez pas le vecteur de gestion des interruptions, par ex. PCINT2 avec interruption réelle, par ex. PCINT20 (PD4). Le mappage du vecteur d'interruption se trouve dans la section 30 page 278 de la fiche technique

3. Interruptions externes dans Arduino

Bien que nous travaillions principalement avec des registres, dans Arduino, les interruptions peuvent également être configurées à l'aide de la fonction **attachInterrupt()** . La fonction reçoit en paramètre le numéro de l'interruption externe (sur Arduino UNO on a INT0, INT1), la fonction qui sera appelée (équivalent à ISR), et la manière dont la transition est détectée :

- BAS : niveau logique 0
- CHANGE : transition de niveau logique
- RISING : front montant
- FALLING : front descendant

La désactivation des interruptions externes peut être effectuée avec la fonction **detachInterrupt()**
L'activation ou la désactivation temporaire de toutes les interruptions peut être effectuée avec les fonctions **noInterrupts()** et **interrupts()**

```
compteur entier =  0 ;

void myCallback ( )  {
  compteur +=  1 ;
}

void setup ( )  {
```

```
  pinMode ( 2 , INPUT_PULLUP ) ;
  attachInterrupt ( 0 , myCallback , FALLING ) ;
}

void loop ( )  {
  Série. print ( "numar apasari: " ) ;
  En série. println ( compteur ) ;
  retard ( 1000 ) ;
}
```

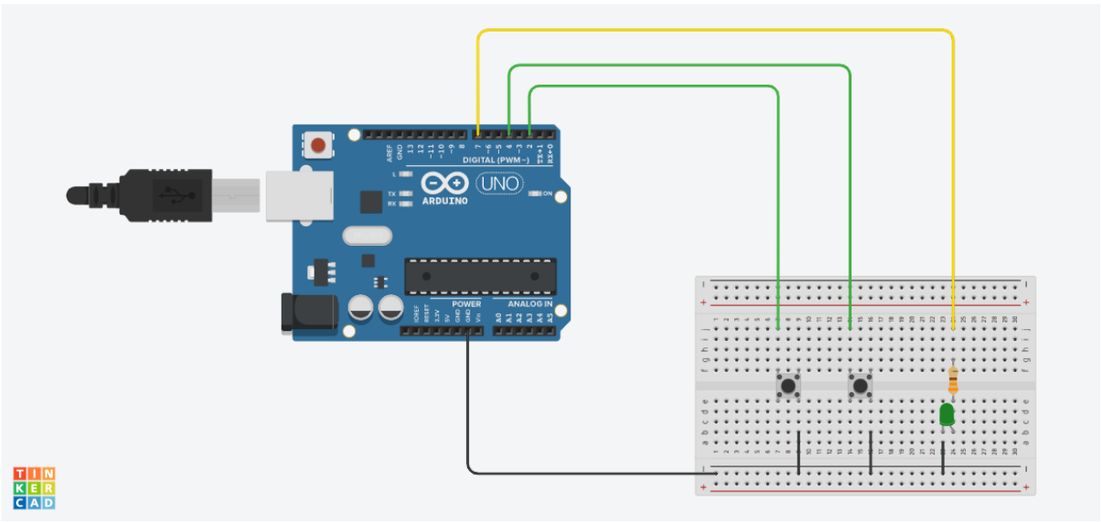
4. Exercices

Tâche 1 (interruptions / boutons) - 6p

Utilisez des interruptions externes (INT et/ou PCINT) pour détecter une pression de bouton connectée à PD2(broche 2) et une autre connectée à PD4(broche 4). Modifiez l'état d'une LED connectée à PD7(broche 7) dans l'ISR.

- Configurer l'interruption externe (INT) sur le front descendant (front descendant) ou le PCINT correspondant à la broche utilisée
- Changer l'état de la LED lorsqu'un bouton est enfoncé (PD2, PD4)

Tinkercad - Testez le programme en utilisant le montage suivant :



Arduino - Téléchargez le code sur la carte et observez le comportement. La LED change-t-elle d'état une seule fois lorsque le bouton est enfoncé ? Ajoutez une méthode anti-rebond [https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce] pour détecter une pression sur un seul bouton :

- Il n'est PAS recommandé d'utiliser le délai dans ISR
- On peut utiliser la fonction *millis()* pour vérifier l'intervalle de temps depuis la dernière pression (100 ms devraient suffire pour un bouton-poussoir/micro-interrupteur, mais ajustez si nécessaire)

```
ISR ( INT0_vect )
{
  // code d'interruption externe
  PORTD ^=  ( 1  << PD7 ) ;
}

ISR ( PCINT2_vect )  {
  // code d'interruption de changement de broche
  // TODO
}

void setup_interrupts ( )  {
  // bouton 1 : PD2 / INT0
  // bouton 2 : PD4 / PCINT20
  cli ( ) ;

  // entrée
  DDRD &= ~ ( 1  << PD2 )  & ~ ( 1  << PD4 ) ;
  // entrée pullup
  PORTD |=  ( 1  << PD2 )  |  ( 1  << PD4 ) ;

  // configurer les interruptions
  // interruptions externes
  // TODO

  // interruptions de type de changement de broche (activer le vecteur d'interruption)
  // TODO

  // activer les interruptions
  // interruption externe
  // TODO

  // interruption de changement de broche
```

```
  // TODO

  être ( ) ;
}

void setup ( )  {
  setup_interrupts ( ) ;
  DDRD |=  ( 1  << PD7 ) ;
  PORTD &= ~ ( 1  << PD7 ) ;
}

boucle vide ( )  {

}
```

Tâche 2 (contrôle du clignotement) - 4p

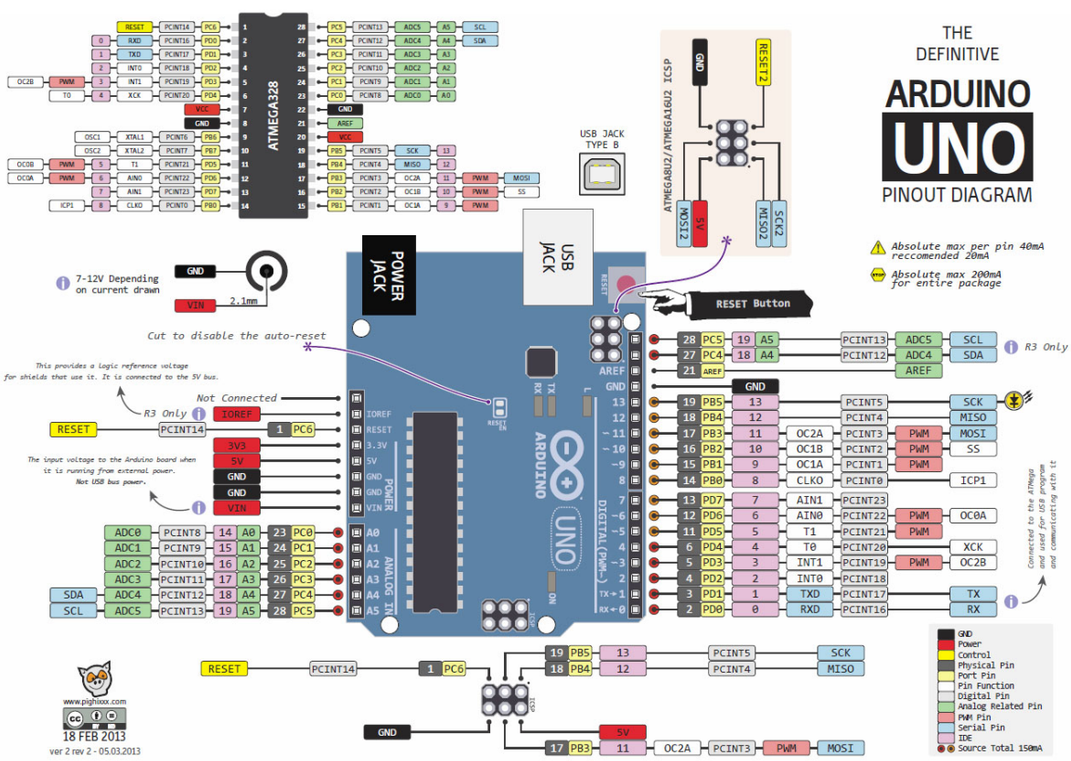
Sur la base du même montage, utilisez des interruptions externes (INT et/ou PCINT) pour détecter l'appui sur un bouton connecté à PD2(broche 2) et un autre connecté à PD4(broche 4). Modifiez l'intervalle de clignotement d'une LED connectée à PD7(broche 7) dans l'ISR comme suit :

- le bouton connecté à PD2augmente l'intervalle de 100ms
- le bouton connecté à PD4diminue l'intervalle de 100 ms

```
entier dt =  500 ;

ISR ( INT0_vect )
{
  // code d'interruption externe
  // TODO
}

ISR ( PCINT2_vect )  {
  // code d'interruption de changement de broche
  // TODO
}

// setup_interrupts() identique à la tâche 1

void setup ( )  {
  setup_interrupts ( ) ;
  DDRD |=  ( 1  << PD7 ) ;
  PORTD &= ~ ( 1  << PD7 ) ;
}

boucle vide ( )  {
  PORTD |=  ( 1  << PD7 ) ;
  retard ( dt ) ;
  PORTD &= ~ ( 1  << PD7 ) ;
  retard ( dt ) ;
}
```

5. Ressources

- Fiche technique Atmega 328p
- Brochage Arduino UNO



- Responsable : Alexandru Predescu [mailto:alexandru.predescu@upb.ro]

5. Liens utiles

- AVR Libc - interruption.h [http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html]
- Jeu d'instructions AVR [http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf]
- MinuterieUn [https://www.arduino.cc/reference/en/libraries/timerone/]
- Ton() [https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/]
- Interruptions [https://www.renesas.com/us/en/support/engineer-school/mcu-programming-peripherals-04-interrupts]