Chapitre 7

Les Interruptions

7.1 Introduction

Un système embarqué et temps réel est nécessairement amené à avoir des interactions avec son environnement externe. L'information échangée lors d'une interaction entre un processeur de traitement et sa périphérie pourra être constituée d'un simple signal, de quelques bits, d'un groupe d'octets, voire d'un bloc complet de caractères(octet), comme par exemple lors d'un transfert de fichiers.

Les échanges ont lieu essentiellement au travers d'interfaces de périphériques d'entrée/sortie. Matériellement, il s'agit d'interfaces programmables ou de contrôleurs d'entrées-sorties. Ces équipements sont plus ou moins complexes, selon leurs fonctionnalités et leurs performances. La vitesse de traitement des équipements périphériques est également très variable. Elle dépend de la nature et de la fonction même de ces équipements périphériques.

Si dans les systèmes à usage général, la gestion des périphériques est effectuée par le système d'exploitation lui-même, dans un système embarqué ou temps réel, cette gestion est fréquemment effectuée directement par des processus dédiés du programme. Cela s'explique notamment par les contraintes temporelles qui ne peuvent être satisfaites qu'au travers d'un couplage serré (couplage fort) entre threads de l'application et threads de gestion à proprement parler de ces équipements.

7.2 Interruption ou Scrutation

L'interaction avec l'environnement externe est réalisés à travers des périphériques tels que: une souris, un écran, un convertisseur analogique numérique, etc. Au niveau de la gestion de l'interaction avec l'environnement périphérique, on relève trois modes usuels.

7.2.1 La scrutation (polling)

La scrutation, polling en anglais, est un mode de communication de nature synchrone, où le processeur scrute périodiquement l'interface afin de savoir si l'équipement est prêt pour une nouvelle opération d'entrée-sortie.

Dans la méthode de scrutation, le microprocesseur est toujours occupé. Soit il est entrain de scruter une entrée, soit il est entrain de faire le traitement d'une entrée déjà scrutée. Cette méthode présente plus d'inconvénients que d'avantages, avec un inconvénient certain: le microprocesseur semble mal conçu, ce qui le force à fonctionner tout le temps, et scruter les périphériques, même si la plupart du temps aucune entrée n'est effectivement disponible.

Considérons une situation, comme représenté sur la figure **??**, dans laquelle le microcontrôleur doit traiter les entrées issues de trois périphériques, D_1, D_2 et D_3.

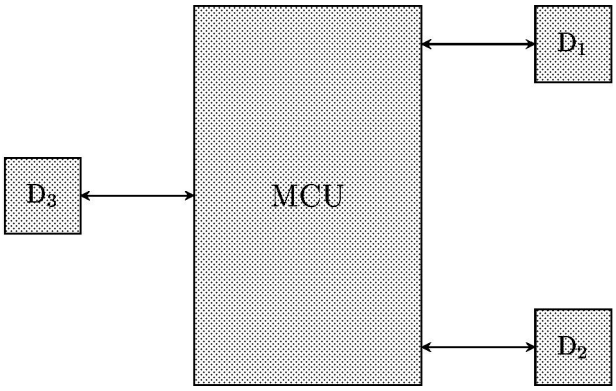


FIGURE 7.1 – Microcontrôleur en interaction avec des dispositifs d'entrée/sortie

L'un des moyens d'assurer le service des trois dispositifs consiste à interroger chaque dispositif pour obtenir des informations selon l'organigramme représenté sur la figure **??**.a.

Un inconvénient évident de la scrutation est qu'il consomme beaucoup de capacité de traitement. Comme tous les périphériques sont interrogés de manière séquentielle, il n'y a pas de mécanisme de priorité explicite dans la méthode de scrutation.

Considérons que D_1 et D_3 ont des entrées pour le microprocesseur (c'est-à-dire qu'ils nécessitent le service du processeur et que le processeur a commencé à travailler sur les entrées de D_1). Ainsi, l'entrée de D_3 doit attendre que le processeur termine son travail avec D_1 , et interroger D_2 avant de pouvoir commencer à traiter l'entrée de D_3 . Même si le périphérique D_3 a une priorité élevée, le microprocesseur ne peut être amené à traiter D_3 avant d'avoir terminé son travail avec D_1 et son interrogation de l'entrée D_2.

Que se passe-t-il si D_3 est un port d'entrée et que sa mémoire tampon risque de

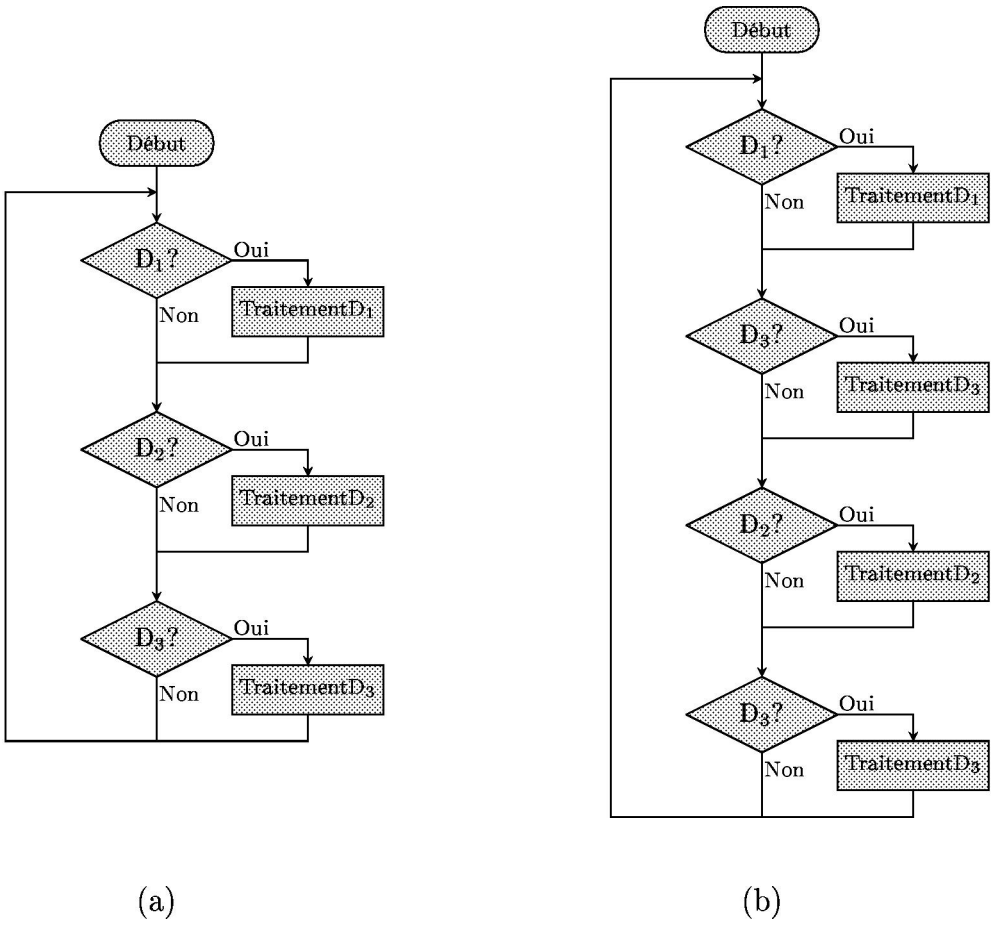


FIGURE 7.2 – Interaction avec des dispositifs d'entrée/sortie par scrutation(polling)

déborder si elle n'est pas lue pendant un certain temps ? Nous pouvons peut-être affiner la boucle et les fonctions qui gèrent D_1 et D_2) de manière à ce que D_3 soit traité avant qu'il ne soit débordé.

Un moyen simple est de modifier la boucle principale, comme décrit dans l'organigramme **??**.b.

Dans ce cas, si le temps de traitement de D_1 et D_2 est inférieur au temps de débordement de D_3, le périphérique D_3 est protégé. Parfois, il peut être nécessaire de régler/optimiser avec précision les fonctions qui traitent les entrées de D_1 et D_2 de manière à ce que l'entrée de D_3 ne soit pas en débordement.

Jusqu'à présent, tout va bien. Il semble maintenant que cela fonctionne sans problème. Mais, que se passe-t-il si un autre périphérique est ajouté ou en raison d'une modification du cahier de charge et des exigences, la priorité d'un appareil change, puis la situation devient très gênante. Tous les réglages sont perdus et nous devons nous lancer dans une autre tentative d'ajuster le code et les priorités. Dans ce cas, le codage se réduit à une

simple jonglerie. Même après un second réglage, il se peut qu'une autre adaptation soit nécessaire lors d'un prochain changement dans le cahier de charge et les exigences.

Les problèmes liés à la méthode de scrutation sont les suivants :

— Le microcontrôleur perd beaucoup de temps en scrutant les états des périphériques en permanence.
— Il est impossible d'attribuer des priorités aux appareils car l'état doit être scruté d'une manière séquentielle, un par un, pour tous les périphériques, indépendamment de leur priorité et de leur importance.
— Cela ralentit le système, car les dispositifs hautement prioritaires doivent attendre leur tour lorsque les périphériques moins prioritaires sont interrogés et servis.

7.2.2 Les interruptions

Pour palier à tous ces problèmes, la deuxième méthode, basée sur une solution matérielle, est plus efficace. Cette méthode est appelée méthode des interruptions. Dans cette méthode, chaque fois qu'une donnée est disponible, le périphérique correspondant envoie un signal d'interruption au microcontrôleur, et en réponse à cela, le microcontrôleur arrête l'activité en cours et traite la requête du périphérique, et ensuite, reprend l'activité régulière. De cette façon, la méthode d'interruption offre les avantages suivants :

— Le temps d'un microcontrôleur est utilisé efficacement car il n'y a pas de perte de temps en continu par une interrogation inutile des périphériques.
— Des priorités peuvent être accordées aux différents dispositifs en fonction de leur importance et le dispositif le plus prioritaire peut être programmé pour interrompre les appareils de priorité inférieure.
— Le système devient plus rapide car les activités les plus importantes sont traitées immédiatement.
— Plus de dispositifs peuvent être desservis (mais un seul dispositif à la fois).
— Le système peut être programmé pour ignorer un ou tous les dispositifs lors de la gestion des tâches critiques.

La source d'interruption peut être des périphériques internes comme des timers, des communications série ou tout autre dispositif externe. Lorsqu'un périphérique génère une interruption, le microcontrôleur est forcé d'appeler un sous-programme associé à ce périphérique; ce sous-programme est plus communément appelé Interrupt Service Routine (ISR) ou Interrupt Handler.

L'avantage évident de cette méthode par rapport à la méthode de scrutation est que le processeur est libre de faire d'autres travaux (exécuter d'autres applications) lorsqu'il n'y a pas d'entrée à ses périphériques.

7.2.3 L'accès direct à la mémoire (DMA: Direct Memory Access)

L'accès direct à la mémoire constitue aussi un mode asynchrone, mais il induit un équipement matériel(hardware) complexe. Ce mode permet de minimiser le nombre d'interruptions. Il est adapté aux transfert directs de blocs d'information entre la mémoire et tout équipement périphérique conforme à ce mode. L'échange de donnée, directement du périphérique à la mémoire, est géré par le DMA sans intervention du microprocesseur.

7.3 Les interruptions

Une interruption est la possibilité offerte par le microprocesseur de communiquer avec l'environnement extérieur pour permettre au microprocesseur de détourner son fonctionnement en fonction de la priorité. Les interruptions peuvent être utilisées pour diverses applications dans des environnements différents.

Une interruption est un processus par lequel un dispositif externe peut attirer l'attention du microprocesseur. Le processus commence à partir du dispositif d'entrée/sortie et il s'agit d'un transfert de données de type asynchrone. La figure 6.1 montre le transfert de données par interruption. Le microprocesseur peut lancer le transfert de données après avoir reçu un signal d'interruption du dispositif d'entrée/sortie. Le microprocesseur peut scanner la broche d'interruption à chaque cycle de la machine. Lorsque le signal d'interruption est présent, le microprocesseur suspend son fonctionnement actuel après avoir stocké l'état actuel dans le microprocesseur afin que le microprocesseur puisse reprendre le travail suspendu là où il l'avait laissé. Par conséquent, la pile est utilisée pour stocker l'état actuel. Ensuite, le microprocesseur assure le service de la demande d'interruption en exécutant une routine de service d'interruption(ISR - Interrupt Service Routine).

Une interruption est considérée comme un signal d'urgence, qui peut être servi. Le microprocesseur peut y répondre dès que possible. Lorsque le microprocesseur reçoit un signal d'interruption, il suspend le programme en cours d'exécution et passe à une routine de service d'interruption (ISR) pour répondre à l'interruption entrante. Chaque interruption aura très probablement son propre ISR. La figure 6.2 montre l'exécution de l'interruption.

La réponse à une interruption peut être immédiate ou retardée selon que l'interruption est masquable ou non et selon que les interruptions sont masquées ou non. Il existe deux façons différentes de rediriger l'exécution vers l'ISR selon que l'interruption est vectrorisée ou non vectorisée. Dans le cas d'une interruption vectorisée, l'adresse du sous-programme est déjà connue du microprocesseur. Dans le cas d'une interruption non vectorisée, les dispositifs d'entrée/sortie devront fournir l'adresse du sous-programme au microprocesseur. Dans tout système à microprocesseur, les dispositifs d'entrée/sortie peuvent utiliser le transfert de données par interruption. Un microprocesseur peut avoir un niveau

d'interruption et plusieurs dispositifs d'entrée/sortie peuvent partager le même niveau d'interruption.

Le microprocesseur peut avoir, aussi, plusieurs niveaux d'interruption et plusieurs dispositifs d'entrée/sortie. Chaque dispositif peut être connecté à un niveau d'interruption. Lorsque plusieurs dispositifs d'entrée/sortie sont connectés à un seul niveau d'interruption, ces dispositifs sont reconnectés à un contrôleur d'interruption tel que le 8259.

Avec un contrôleur d'interruption 8259, seuls huit dispositifs d'entrée/sortie peuvent être connectés. Lorsque plus de huit dispositifs d'entrée/sortie sont connectés, plusieurs contrôleurs d'interruption sont connectés en cascade.

Si chaque dispositif E/S est connecté à un niveau d'interruption indépendant, le microprocesseur doit avoir plusieurs niveaux d'interruption. Dans ce cas, le nombre de dispositifs d'entrée/sortie doit être inférieur au nombre de niveaux d'interruption.

Les interruptions, comme leur nom l'indique, interrompent le déroulement d'un programme en cours pour exécuter Interrupt Service Routine(ISR) qui fait tout ce qui est nécessaire lorsque l'interruption se produit. Les interruptions sont utiles dans les situations où le processeur doit répondre immédiatement à l'interruption ou dans les cas où il est très coûteuse pour le processeur pour vérifier l'occurrence d'un événement. Les exemples de la nécessité d'une réponse immédiate comprennent l'utilisation des interruptions pour suivre l'heure (l'interruption peut permettre de mettre à jour une horloge) ou un bouton d'arrêt d'urgence qui arrête immédiatement une machine lorsqu'une urgence se produit.

Parmi les applications utiles des interruptions, citons les dispositifs tels que les claviers ou les autres dispositifs d'entrée. Il serait inutile que le processeur interroge un clavier dans l'espoir qu'on a appuyé sur une touche. Dans ce cas, la pression d'une touche peut provoquer une interruption et le microcontrôleur faisait alors une brève pause pour voir si la pression sur la touche l'exigeait pour faire quoi que ce soit. Si la pression sur la touche était la première de plusieurs requises pour provoquer une action, le processeur reprenait ses tâches jusqu'à ce que suffisamment de touches aient été actionnées pour exiger une action. L'utilisation d'interruptions libère le processeur de l'obligation d'interroger le clavier en permanence à un rythme qui est beaucoup, beaucoup plus vite que les appuis sur les touches.

Définition: Une interruption est un moyen de faire une requête de manière asynchrone au processeur pour exécuter un service désiré.

— **Asynchrone :** parce que l'interruption peut provenir d'un autre dispositif qui n'est pas nécessairement synchronisé par l'horloge du système. L'interruption peut arriver indépendamment de l'horloge du microprocesseur. (Cependant, les interruptions ne sont présentées au microprocesseur que de manière synchrone - le traitement est synchrone avec l'horloge du système).

— **Requête :** Les interruptions (sauf celle appelée interruption non masquable) sont des requêtes. Le processeur est libre de différer le service de l'interruption lorsqu'il travaille sur un code de priorité plus élevée (cela pourrait être la routine de service pour une interruption de priorité plus élevée).
— **Exécuter le service désiré :** Un périphérique interrompt le microprocesseur pour lui demander d'effectuer un service défini ou de transmettre certaines données au processeur.

7.4 Classification des interruptions

Les interruptions peuvent être classées selon plusieurs critères. Les interruptions peuvent être classées en interruptions vectorisées et interruptions non vectorisées.

— **Les interruptions vectorisées**: Dans les interruptions vectorisées, l'adresse de la routine de service d'interruption est câblée;
— **Les interruptions non vectorisées**: L'adresse de la routine de service doit être fournie de l'extérieur par l'appareil envoyant la requête d'interruption.

Les interruptions peuvent être classées en fonction de la source de l'interruption.

— **Interruptions matérielles :** Si le microprocesseur est interrompu par un dispositif/matériel externe, alors l'interruption peut être classée comme une interruption matérielle.
— **Interruptions logicielles :** Elles peuvent être déclenchées en exécutant des instructions spéciales du microprocesseur. Les interruptions logicielles peuvent être classées comme des interruptions ou des exceptions. Si une interruption est planifiée (ou se produit de manière déterministe, ou si elle est prévue ou provoquée intentionnellement), elle peut être classée comme une interruption logicielle. Si elle n'est pas planifiée, elle est alors classée comme une exception.

On distingue quelquefois deux types d'interruptions matérielles

— **Les interruptions non masquables:** Les interruptions non masquables ne peuvent pas être retardées ou ignorées. Elles sont en général réservées aux fonctions critiques du système, par exemple sauver l'état du microprocesseur en cas de coupure imminente du courant.
— **Interruptions masquables:** Ces interruptions peuvent être retardées ou ignorées par des interruption de priorité plus élevée.

Il existe encore une autre façon de classer les interruptions en fonction de leur périodicité:

— **Interruptions périodiques :** si les interruptions se produisent à intervalles fixes dans le temps, on peut alors les qualifier de périodiques.
— **Interruptions apériodiques :** si les interruptions peuvent se produire à tout

moment, alors on les appelle des interruptions apériodiques. (par exemple, pour anticiper une pression de touche de l'utilisateur).

Les interruptions peuvent également être classées en fonction de leur relation temporelle avec l'horloge système du processeur:

- **Interruptions synchrones :** si la source de l'interruption est alignée exactement en phase avec l'horloge système (l'horloge utilisée par le microprocesseur), alors la source de l'interruption est dite synchrone (Fig.xxxx). Par exemple, une interruption timer qui utilise l'horloge du système.
- **Interruptions asynchrones :** si la source d'interruption n'est pas en phase avec le système elle est qualifiée d'asynchrone (Fig. 5.4).

Cela peut sembler contradictoire avec la définition de l'interruption, mais la définition couvre les interruptions matérielles qui proviennent des dispositifs externes. Les interruptions synchrones se produisent à cause des interruptions logicielles ou des dispositifs qui sont pilotés par l'horloge système et forment généralement un petit sous-ensemble d'interruptions provenant de dispositifs externes dans un système embarqué typique.

7.5 Vecteur d'interruption

Lorsqu'une interruption survient, le microprocesseur a besoin de connaître l'adresse de la routine de service de cette interruption. Pour cela, la source d'interruption place sur le bus de données un code numérique indiquant la nature de l'interruption. Le microprocesseur utilise ce code pour rechercher dans une table en mémoire centrale l'adresse du sous-programme d'interruption à exécuter. Chaque élément de cette table s'appelle un vecteur d'interruption. Lorsque les adresses des sous-programmes d'interruptions sont gérées de cette manière, on dit que les interruptions sont vectorisées. L'avantage de la vectorisation des interruptions est que l'emplacement de la routine (ISR) peut être n'importe où dans la mémoire, il suffit de spécifier le vecteur d'interruption correspondant.

7.6 Temps de latence des interruptions

Le temps de latence est la durée qui sépare la réception d'une interruption matérielle et l'exécution du code associé à cette interruption. Plus la latence est grande, plus le temps de réponse est important. Ainsi, l'utilisateur aura le sentiment que le système n'est pas assez réactif.

7.7 Priorité des interruptions et contrôleur programmable des interruptions

Les microprocesseurs et les microcontrôleurs n'ont généralement que quelques broches d'interruption. Ce nombre est généralement compris entre deux et quatre. Ce nombre est normalement inférieur au nombre de dispositifs avec lesquels que le processeur doit s'interfacer. Ainsi, dans une situation où plusieurs dispositifs doivent interrompre le microprocesseur, un matériel appelé contrôleur programmable d'interruption (programmable interrupt controller) est utilisé.

Le contrôleur programmable d'interruption peut être utilisé pour attribuer différents niveaux de priorité aux interruptions différentes . L'appareil se charge également de fournir des informations afin d'acheminer l'interruption vers sa routine de service.

La distribution de la logique d'interruption dans chaque unité est très intéressante par son aspect modulaire, mais elle conduit à une quantité relativement importante de matériel dans son interface, et convient mal lorsque les interfaces sont simples et nombreuses. Une variante consiste à concentrer toute la logique d'interruption dans une interface unique, dotée d'une ligne de demande d'interruption pour chacune des autres interfaces.

Ce contrôleur d'interruption est en fait une interface programmable particulière, comportant les registres nécessaires au masquage des interruptions et à la génération du vecteur d'interruption. Une logique fixe la priorité des demandes, selon une règle qui peut être programmable.

Du point de vue programmation, le contrôleur d'interruption est plus facile à gérer qu'un ensemble de bits de commande d'interruption répartis dans des périphériques, mais les liaisons en étoile depuis le processeur sont contraires à la notion de bus formé de lignes parallèles et de connecteurs tous équivalent.

7.8 Masquage des interruptions

7.9 Interrupt service routines

Une interruption, comme son nom l'indique, consiste à interrompre momentanément le programme en exécution pour effectuer un autre travail. Quand cet autre travail est terminé, le microprocesseur retourne à l'exécution du programme et reprend à l'endroit exact où il l'avait laissé. Cet autre travail s'appelle le programme d'interruption ou la routine d'interruption ou encore une ISR pour Interrupt Service Routine en anglais.

Que se passe-t-il si une nouvelle interruption survient alors que l'ISR déclenchée par la précédente n'est pas terminée ? Une ISR n'est pas interrompue par une nouvelle in-

terruption. La nouvelle interruption ne sera prise en compte que lorsque l'ISR en cours se terminera. Le corollaire est qu'il ne faut pas appeler de fonctions qui se mettent en attente d'une interruption à partir d'une ISR. Comme l'interruption attendue ne peut pas déclencher une nouvelle ISR, la fonction attendra indéfiniment et tout le système se bloquera. C'est ce que l'on appelle un deadlock.

Que se passe-t-il si plusieurs interruptions surviennent en même temps ? Les Interruptions ont chacune une priorité. Par exemple, les interruptions externes sont plus prioritaires que les interruptions des Timers. Le MCU exécutera les ISR dans leur ordre de priorité. L'ordre de priorité est donné dans la table ci-dessous. La source d'interruption située la plus en haut de la table est la plus prioritaire. Il est très important que les ISR aient un temps d'exécution le plus court possible. On ne fera donc aucun calcul compliqué et aucun appel à des fonctions longues comme un affichage sur un écran LCD.

Les fonctions de Serial qui permettent d'afficher, via la ligne série et l'USB dans le moniteur série font exactement cela. Leur appel à partir d'une ISR est donc interdit.

Lors d'une interruption, le processeur enregistre tout ou une parte de son état interne, généralement dans la pile système, et exécute ensuite une routine d'interruption , généralement en suivant les directives d'une table indiquant pour chaque interruption, le sous programme à exécuter.

Une fois le traitement de l'interruption terminé, la routine se finit normalement par une instruction de retour d'interruption, qui restaura l'état enregistré et fait repartir le processeur de l'endroit où il avait été interrompu.

Lors du fonctionnement de certaines parties du système d'exploitation, il peut être nécessaire d'interdire les interruptions, soit parce que celles-ci perturberaient un compte serré du temps, soit parce que des structures de données sont en cours de modification (on réalise ainsi une sorte de verrou d'exclusion mutuelle dans un système mono-processeur). Aussi, on peut généralement bloquer (on dit souvent masquer) les interruptions. Dans la plupart des systèmes, les interruptions bloquées sont accumulées, c'est-à-dire qu'elles sont exécutées dès qu'elles sont démasquées. Cependant, pour chaque type d'interruption, le compteur d'interruptions en attente se réduit souvent à un simple drapeau ; si cela peut ne pas être gênant si l'interruption signale des données en attente sur un périphérique, cela peut cependant occasionner des mauvais comptes si l'interruption déclenche l'incrémentation d'une horloge, si les interruptions sont bloquées pour une durée supérieure à la période de l'horloge.

Sur certains systèmes, il existe une interruption non masquable, généralement dédiée au signalement d'une erreur catastrophique pour le système (par exemple, détection d'une erreur mémoire par code correcteur d'erreurs).

Les interruptions peuvent par ailleurs être hiérarchisées suivant des priorités. Une in-

terruption de priorité supérieure est prise en compte lors de la fin du traitement d'une autre interruption, mais une interruption de priorité inférieure est mise en attente. Classiquement une interruption est provoquée par une source extérieure au processeur comme un périphérique. Le processeur doit donc être capable de différencier les différentes sources pour réaliser le traitement voulu. Pour gérer cela, le processeur dispose d'un gestionnaire d'interruption dont le rôle est d'associer à chaque interruption sa fonction de traitement.

Basiquement, une interruption est un signal qui interrompt l'activité normale du microcontrôleur. Elle peut être d'origine interne (changement d'état d'une broche) ou interne (compteur, fin de transmission, etc). Lorsqu'elle est déclenchée, une interruption met en pause le programme principal pour exécuter un bloc de code, qu'on appelle une routine d'interruption (en anglais Interrupt Service Routine, ISR). Une fois cette routine effectuée, le programme reprend là où il s'était arrêté.

Pourquoi se compliquer la vie juste pour répondre à des événements particuliers alors que vous savez déjà vérifier l'état d'une broche et créer vos propres chronomètres ? L'avantage des interruptions est de vous permettre de gérer ces événements de manière asynchrone, c'est-à-dire à n'importe quel moment du programme, indépendamment de son déroulement normal. Ce qui revient à laisser le microconrôleur faire le travail tout seul plutôt que de vérifier manuellement si l'événement attendu est bien arrivé.

Les microcontrôleurs ne peuvent pas réaliser plus d'une tôche à la fois. Gérer certains événements de façon asynchrone via les interruptions permet d'optimiser le code, en évitant de gaspiller de précieux cycles d'horloge à faire des lectures répétées ou à attendre que quelque chose arrive. Les interruptions sont aussi utiles pour les applications qui nécessitent un timing précis, parce que l'on est sûr de détecter l'événement au moment où il a lieu, et de ne rien rater. Pour utiliser le mécanisme d'interruption en provenance d'un périphérique, il va falloir :

— autoriser le périphérique à lancer une interruption;
— configurer le processeur pour gérer l'interruption ;
— écrire le code associé à l'interruption.

Chaque microcontrôleur AVR a une liste de vecteurs d'interruptions, c'est-à-dire d'événements qui peuvent déclencher une interruption. Lorsque celles-ci sont autorisées et que l'un des événements en question a lieu, le code va faire un saut à un endroit spécifique de la mémoire : l'adresse du vecteur d'interruption. En écrivant une routine et en la liant à l'adresse mémoire du vecteur d'interruption, on indique à notre programme quel bloc de code spécifique exécuter lorsque l'interruption est appelée.

Pour utiliser correctement une interruption, nous devons faire trois choses : Activer le bit d'activation globale des interruptions (Global Enable Interrupts) dans le registre de statut (Status Register) de l'AVR Activer le bit d'activation spécifique à l'interruption

qui nous intéresse (chaque vecteur a son propre on/off). Ecrire une routine et l'attribuer à notre vecteur d'interruption cible.

7.10 Gestion des interruptions

Si des interruptions sont utilisées dans un programme, Il est nécessaire de:

— Configurer la ou les ressources pour suivre le ou les événements.

— Activer la ou les interruptions (individuelles et globales, dans chaque cas).

— Poursuivre l'exécution normale de la requête.

Dans les applications qui utilisent des interruptions, il est courant de faire boucler le programme principal dans une boucle infinie sans faire aucune activité, ainsi, le MCU reste inactif, laissant la fonctionnalité du système aux ISR.

Lorsqu'une interruption se produit, le microcontrôleur exécute automatiquement les étapes suivante:

— Acceptation (ou rejet) de la demande d'interruption par l'unité central,

— Termine l'instruction en cours d'exécution.

— Désactive toute les interruptions, de sorte qu'il ne peut pas recevoir une nouvelle interruption tout en s'occupant de celle en cours.

— Sauvegarde l'état du système, c'est-à-dire le contenu des divers registres (compteur ordinal, registre d'état, ...), dans la pile, de manière à pouvoir reprendre l'exécution du programme interrompu en l'état où il se trouvait au moment de l'interruption,

— Attribue au compteur programme le vecteur d'interruption correspondant à l'interruption active pour passer à l'ISR.

— Exécute l'ISR. Lorsque l'exécution de l'ISR est terminée, dans le MCU:

— Le fag associé à l'événement qui a généré l'interruption est effacé.

— Les interruptions sont activés.

— Le compteur ordinal est chargé par l'adresse sauvegardée dans la pile et l'état dans le quel se trouvait le système au moment de la prise en compte de l'interruption est restauré, et l'exécution du programme principal se poursuit.

7.11 Règles d'écritures des ISRs

7.12 Les interruptions de l'Atmega328p

Les microcontrolleurs AVR ont une structure d'interruption riche. xxxx Les différentes sources interruptions de l'Atmega328p sont:

— Les interruptions liées aux entreés:INT0(PD2) et INT1(PD3),

— Les interruptions sur changement d'état des broches: PCINT0 à PCINT23,
— Les interruptions liées au timers 0,1 et 2(débordement),
— Les interruption liée au comparateur analogique,
— L'interruption de fin de conversion ADC,
— Les interruptions du port série USART,
— Les interruptions du bus I2C.

Toutes les interruptions, y compris RESET, fonctionnent exactement de la même manière. Les interruptions sont vectorisées, chaque interruption a une adresse vectorielle qui lui est assignée dans la mémoire programme. Le compilateur place l'adresse de départ de la routine de service d'interruption associée et une instruction de saut relative à la position du vecteur d'interruption. Lorsque l'interruption se produit, le programme termine l'exécution de l'application en cours et se branche ensuite à l'emplacement du vecteur associé à cette interruption. Le programme exécute alors l'instruction de saut relatif vers l'ISR et commence l'exécution de la routine d'interruption.

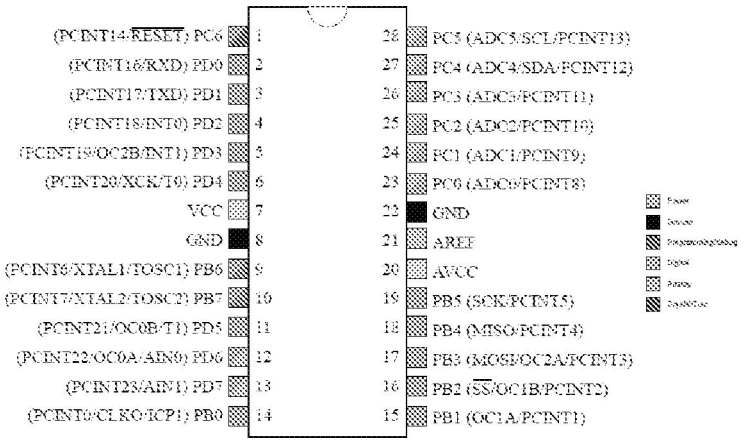


FIGURE 7.3 – Brochage du microcontrôleur Atmaega328p

Lorsque une interruption se produit, l'adresse de retour (l'adresse de la prochaine instruction à exécuter lorsque l'interruption est terminée) est stockée dans la pile du système. La dernière instruction de la routine d'interruption du service est une instruction RETI en langage assembleur. RETI est un retour d'une interruption(RETurn from Interrupt). Cette instruction provoque l'extraction de l'adresse de retour de la pile et la poursuite de l'exécution du programme à partir du moment où l'exécution a été interrompue.

Dans les processeurs AVR, toutes les interruptions ont la même priorité. Il n'y a pas de possibilité q'une interruption interrompt une autre interruption, c'est-à-dire qu'une interruption ne peut pas être prioritaire q'une autre interruption. Il est toutefois possible que deux interruptions se produisent simultanément. Un arbitrage parfois appelé priorité,

est prévu pour déterminer quelle est l'interruption à exécuter dans ces cas. Dans le cas où deux interruptions se produisent simultanément, l'interruption avec le vecteur le plus faible sera exécutée en premier.

La séquence des évènements lorsque une interruption survient sont les suivants:

— Le dispositif périphérique interrompt le processeur,
— L'instruction courante est complétée,
— L'adresse de l'instruction suivante est stockée dans la pile,
— L'adresse de la subroutine d'interruption (ISR) est chargée dans le compteur programme,
— Le processeur exécute la subroutine d'interruption,
— La fin de l'exécution de la subroutine d'interruption est indiquée par l'instruction RETI(return from interrupt),
— Le processeur charge le compteur programme par la l'adresse stockée dans la pile et continue l'exécution du programme interrompu.

Les interruptions doivent être initialisées avant de devenir actives ou utilisables. L'initialisation des interruptions est un processus en deux étapes : La première étape consiste à débloquer les interruptions qui doivent être actives, et la deuxième étape consiste à permettre globalement toutes les interruptions non masquées.

Le tableau **??** présente les sources d'interruption disponibles dans microcontrôleur un ATMega328.

7.12.1 Les interruptions externes INT0 et INT1

Les interruptions externes peuvent être déclenchées par un front descendant ou montant ou par un niveau bas. La configuration se fait par le registre de contrôle des interruptions externes A (External Interrupt Control Register A, EICRA). Lorsque les interruptions sont activées et sont configurées comme étant déclenchées par le niveau, les interruptions se déclenchent tant que la broche correspondante est maintenue au niveau bas.

Notez que la reconnaissance des interruptions de front descendant ou montant sur l'INT nécessite la présence d'un E/S horloge. L'interruption de bas niveau sur l'INT est détectée de manière asynchrone. Cela implique que cette interruption peut être utilisée pour le réveil également des modes de sommeil autres que le mode idle. L'horloge E/S est arrêtée dans tous les modes de veille sauf en mode idle.

Remarque: Si une interruption déclenchée par un niveau est utilisée pour le réveil à partir de la mise hors tension, le niveau requis doit être maintenu suffisamment longtemps pour que le MCU puisse terminer le réveil afin de déclencher l'interruption de niveau. Si le niveau disparaît avant la fin du temps de démarrage, l'unité MCU se réveillera quand

Vect.	Adresse	Source	Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out, Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2_COMPB	Timer/Coutner2 Compare Match B
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow
11	0x0014	TIMER2_CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1_COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1_COMPB	Timer/Counter1 Compare Match B
14	0x001A	TIMER1_OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0_COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0_COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0_OVF	Timer/Counter0 Overflow
18	0x0022	SPI STC	SPI Serial Transfer Complete
19	0x0024	USART_RX	USART Rx Complete
20	0x0026	USART_UDRE	USART Data Register Empty
21	0x0028	USART_TX	USART Tx Complete
22	0x002A	ADC ADC	Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002A	ANALOG COMP	Analog Comparator
25	0x0030	TWI 2-wire	Serial Interface (I2C)
26	0x0032	SPM READY	Store Program Memory Ready

TABLE 7.1 – Les interruptions du microcpntrôleur ATmega328

même, mais aucune interruption ne sera générée. Le temps de démarrage est définie par les fusibles SUT et CKSEL.

7.12.2 Les interruptions externes PCINT0, PCINT1 et PCINT2

Ces interruptions externes offrent moins de possibilités que celles que nous avons vues au paragraphe précédent. Tout d'abord, les 3 sources PCINT0, PCINT1 et PCINT2 sont partagées par plusieurs broches. PCINT0 correspond aux interruptions externes sur les broches 8 à 13, PCINT1 aux interruptions externes sur les broches A0 à A5 et PCINT3 aux interruptions externes sur les broches 0 à 7. On peut noter que les broches 2 et 3 sont donc partagées entre PCINT3 et INT0 et INT1. On ne peut donc à la fois bénéficier de INT0 et INT1 et des interruptions PCINT3 sur les broches 2 et 3. Comme la même ISR est exécutée pour toutes les broches qui partagent sa source d'interruption, la discrimination de la broche qui a engendré l'interruption, ou des broches en cas de changements simultanés, doit être faite dans la routine d'interruption.

Ensuite, la richesse des modes de déclenchement de INT0 et INT1 n'existent pas pour les sources PCINTx. Le seul mode est l'interruption sur changement d'état, LOW vers HIGH et HIGH vers LOW. Par conséquent si seul l'un des changements d'état intéresse le programme, il faut également discriminer en allant lire l'état de la broche qui a déclenché l'interruption.

Les interruptions externes sont déclenchées par les broches INTx ou l'une des broches PCINTx. Il faut noter que, si elles sont activées, les interruptions externes sont déclenchées par les broches INTx ou l'une des broches PCINTx, les interruptions se déclenchent même si les broches INT ou PCINT sont configurées comme des sorties. Cette fonction permet de générer des interruptions logicielles.

La broche Pin Change Interrupt Request 2 (PCI2) se déclenchera si une broche PCINT[23:16] active change d'état. La broche Change Interrupt Request 1 (PCI1) se déclenchera si une broche PCINT[14:8]active change d'état. La broche Pin Change Interrupt Request 0 (PCI0) se déclenchera si une broche PCINT[7:0] active change d'état.

Les registres PCMSK2, PCMSK1 et PCMSK0 contrôlent les broches qui contribuent aux interruptions de changement d'état. Les interruptions de changement d'état dans PCINT sont détectés de manière asynchrone. Cela implique que ces interruptions peuvent être utilisées pour le réveil également à partir de mode de sommeil autres que le mode idle.

Activation des interruptions externes

Toute interruption n'aura lieu que si l'interruption globale et l'interruption individuelle à utiliser ont été activées. L'activation globale des interruption se fait par la mise à 1 du bit I, situé à la position 7 dans le registre d'état (SREG). Chaque interruption doit être aussi activé dans le registre dédié; le registre External Interrupt Mask Register (EIMSK) pour les interruption INT0 et INT1 et les registres Pin Change Mask Register 0, Pin Change Mask Register 1 et Pin Change Mask Register 3 pour les interruptions PCINT.

Configuration et état des interruptions externes

Trois registres gèrent les interruptions externes INT0 et INT1. Le registre EICRA (External Interrupt Control Register A), le registre External Interrupt Mask Register (EIMSK) et le register EIFR(External Interrupt Flag Register). La configuration des interruptions INT0 et INT1 se fait dans le registre EICRA (External Interrupt Control Register A); deux bits sont utilisés, pour chaque interruption, pour configurer le mode de déclenchement des interruptions. L'activation des interruptions est réalisée par les deux bits les moins significatifs du registre External Interrupt Mask Register (EIMSK). Enfin, l'état des interruption est indiqués par les deux bits INTF0 et INTF1 du registre EIFR(External Interrupt Flag Register) qui sont activés par les interruption INT0 et INT1. L'état des interruptions PCINTx est indiqué par les flags correspondants des registres Pin Change Interrupt Flag 0, Pin Change Interrupt Flag 1 et Pin Change Interrupt Flag 2.

7.12.3 Description des registres

Registre d'état - Status Register(SREG)

Le registre d'état (SREG) contient les information sur l'opération arithmétique ou logique qui vient d'être exécutée. Ces informations peuvent être utilisées pour changer le déroulement du programme afin d'exécuter des opérations conditionnelles. Le registre d'état est mis à jour après chaque opération de l'ALU.

Le registre d'état n'est automatiquement ni sauvegardé, ni restitué lors de l'exécution d'une routine d'interruption. Cette opération devra donc être gérée par le programme. Le registre d'état possède huit bits accessibles en lecture et en écriture.

- **Bit 7 - I: Global Interrupt Enable** ce bit à pour rôle de valider les interruptions. Il devra être positionné à 1; si ce bit est à 0, aucune interruption ne pourra s'exécuter. Il faut cependant noter que chaque type d'interruption possède aussi ses propres bits de contrôle dans un registre qui lui est dédié. Le bit I est effacé après le déclenchement d'une interruption, et il est mis à 1 par l'instruction RETI

Bit	7	6	5	4	3	2	1	0
0x3F (0x5F)	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

FIGURE 7.4 – Registre d'état (SREG))

pour permettre l'activation des interruptions postérieures. Le bit I peut également être activé et effacé par l'application des instructions SEI et CLI respectivement.

- **Bit 6 - T: Bit Copy Storage**: ce bit utilise les instructions du langage machine(BLD - BitLoaD et BST - Bit STore), pour effectuer des opérations de lecture et de stockage des bits dans les registres du microcontrôleur.
- **Bit 5 - H: Half Carry Flag** bit de demi-retenue: il indique la présence d'une demi-retenue lors d'une opération arithmétique, notamment sur des nombres BCD. Dans ce cas il prend la valeur 1.
- **Bit 4 - S: Sign Bit, S = N $\oplus$ V** bit de signe: le bit S est positionné à 1, lorsque le bit N vaut 1, ou le bit V vaut 1 de façon exclusive.
- **Bit 3 - V: Two's Complement Overflow Flag** bit de débordement du complément à deux: indique un débordement lors d'une opération arithmétique en complément à deux. Dans ce cas il prend la valeur 1.
- **Bit 2 - N: Negative Flag** ce bit est positionné à 1 lorsque le résultat d'une opération arithmétique ou logique est négatif.
- **Bit 1 - Z: Zero Flag** bit de zéro: indique un résultats nul lors d'une opération arithmétique ou logique. Dans ce cas il prend la valeur logique 1.
- **Bit 0 - C: Carry Flag** bit de retenue: indique la présence d'une retenue lors d'une opération arithmétique ou logique. Dans ce cas il prend la valeur logique 1.

External Interrupt Control Register A(EICRA)

Le registre External Interrupt Control Register A comporte les bits de contrôle pour le contrôle de la présence d'interruptions.

- **Bits 3: - ISC1n: Interrupt Sense Control 1 [n = 1:0]**
  L'interruption externe INT1 est activée par la broche externe INT1 si le flag SREG-I et le masque d'interruption correspondant sont activés. Le niveau et les flancs de la broche externe INT1 qui activent l'interruption sont définis dans le tableau ci-dessous. La valeur sur la broche INT1 est échantillonnée avant de détecter les flancs. Si l'interruption du flanc ou de basculement d'état est sélectionnée, les impulsions

qui durent plus d'une période d'horloge génèrent une interruption. Les impulsions plus courtes ne sont pas garanties de générer une interruption. Si l'interruption de niveau bas est sélectionnée, le niveau bas doit être maintenu jusqu'à la fin de l'instruction en cours d'exécution pour générer une interruption.

Valeur	Description
00	Un niveau bas du INT1 génère une requête d'interruption.
01	Tout changement d'état logique de INT1 génère une requête d'interruption.
10	Un flanc descendant de INT1 génère une requête d'interruption.
11	Un flanc montant de INT1 génère une requête d'interruption.

TABLE 7.2 – Les interruptions du microcpntrôleur ATmega328

- **Bits 1:0 - ISC0n: Interrupt Sense Control 0 [n = 1:0]**
  L'interruption externe INT1 est activée par la broche externe INT1 si le flag SREG-I et le masque d'interruption correspondant sont activés. Le niveau et les flancs de la broche externe INT1 qui activent l'interruption sont définis dans le tableau ci-dessous. La valeur sur la broche INT1 est échantillonnée avant de détecter les flancs. Si l'interruption du flanc ou de basculement d'état est sélectionnée, les impulsions qui durent plus d'une période d'horloge génèrent une interruption. Les impulsions plus courtes ne sont pas garanties de générer une interruption. Si l'interruption de niveau bas est sélectionnée, le niveau bas doit être maintenu jusqu'à la fin de l'instruction en cours d'exécution pour générer une interruption.

Valeur	Description
00	Un niveau bas du INT1 génère une requête d'interruption.
01	Tout changement d'état logique de INT1 génère une requête d'interruption.
10	Un flanc descendant de INT1 génère une requête d'interruption.
11	Un flanc montant de INT1 génère une requête d'interruption.

TABLE 7.3 – Les interruptions du microcpntrôleur ATmega328

Bit	7	6	5	4	3	2	1	0
0x69					ISC11	ISC10	ISC01	ISC00
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

FIGURE 7.5 – External Interrupt Control Register A(EICRA)

External Interrupt Mask Register(EIMSK)

- **Bit 1 - INT1: External Interrupt Request 1 Enable:**
  Lorsque le bit INT1 est mis à 1 et que le bit I dans le registre d'état (SREG) est activé, l'interruption de la broche externe est activée. Les bits ISC11 et ISC10 de la commande Interrupt Sense Control1 dans External Interrupt Control Register A (EICRA) définit si l'interruption externe est activée sur le front montant et/ou descendant de la broche INT1 ou niveau détecté. L'activité sur la broche entraînera une demande d'interruption même si INT1 est configurée comme une sortie. L'interruption correspondante de la requête d'interruption externe 1 est exécutée à partir du Vecteur d'interruption INT1.
- **Bit 1 - INT1: External Interrupt Request 0 Enable :**
  Lorsque le bit INT0 est activé et que le bit I dans le registre d'état (SREG) est activé, l'interruption de la broche externe est activée. Les bits ISC00 et ISC01 de la commande Interrupt Sense Control 0 dans External Interrupt Control Register A (EICRA) définit si l'interruption externe est activée sur le front montant et/ou descendant de la broche INT0 ou niveau détecté. L'activité sur la broche entraînera une demande d'interruption même si la broche INT1 est configurée comme une sortie. L'interruption correspondante de la requête d'interruption externe 0 est exécutée à partir du Vecteur d'interruption INT0.

Bit	7	6	5	4	3	2	1	0
0x3D							INT1	INT0
Access							R/W	R/W
Reset							0	0

FIGURE 7.6 – External Interrupt Mask Register (EIMSK)

External Interrupt Flag Register (EIFR)

- **Bit 1 - INTF1: External Interrupt Flag 1:** Lorsqu'un front ou un changement de niveau sur la broche INT1 déclenche une requête d'interruption, INTF1 sera activé. Si le bit I dans le registre d'état SREG et le bit INT1 dans EIMSK sont activés, l'unité MCU passera au vecteur d'interruption correspondant. Le drapeau est effacé lorsque la routine d'interruption est exécutée. Il est également possible d'effacer l'indicateur en y inscrivant 1. Ce drapeau est toujours effacé lorsque l'interruption INT1 est configurée comme une interruption de niveau.

- **Bit 1 - INTF1: External Interrupt Flag 0:** Lorsqu'un front ou un changement de niveau sur la broche INT0 déclenche une requête d'interruption, INTF0 sera activé. Si le bit I dans le registre d'état SREG et le bit INT0 dans EIMSK sont activés, l'unité MCU passera au vecteur d'interruption correspondant. Le drapeau est effacé lorsque la routine d'interruption est exécutée. Il est également possible d'effacer l'indicateur en y inscrivant 1. Ce drapeau est toujours effacé lorsque l'interruption INT0 est configurée comme une interruption de niveau.

Bit	7	6	5	4	3	2	1	0
0x3C							INTF1	INTF0
Access							R/W	R/W
Reset							0	0

FIGURE 7.7 – External Interrupt Flag Register (EIFR)

Pin Change Interrupt Control Register (PCICR)

- **Bit 2 PCIE2: Pin Change Interrupt Enable 2** Lorsque le bit PCIE2 est mis à un et que le bit I du registre d'état (SREG) est activé, l'interruption de changement de niveau de la broche 2 est activée. Tout changement sur une broche PCINT23..16 activée entraînera une interruption. L'interruption correspondante de la demande d'interruption de changement de niveau est exécutée à partir du vecteur d'interruption PCI2. Les interruptions PCINT23..16 sont activées individuellement par le registre PCMSK2.
- **Bit 1 - PCIE1: Pin Change Interrupt Enable 1** Lorsque le bit PCIE1 est mis à un et que le bit I du registre d'état (SREG) est activé, l'interruption de changement de niveau de la broche 1 est activée. Tout changement sur une broche PCINT14..8 activée entraînera une interruption. L'interruption correspondante de la demande d'interruption de changement de niveau est exécutée à partir du vecteur d'interruption PCI1. Les interruptions PCINT14..8 sont activées individuellement par le registre PCMSK1.
- **Bit 0 - PCIE0: Pin Change Interrupt Enable 0** Lorsque le bit PCIE0 est mis à un et que le bit I du registre d'état (SREG) est activé, l'interruption de changement de niveau de la broche 0 est activée. Tout changement sur une broche PCINT7..0 activée entraînera une interruption. L'interruption correspondante de la demande d'interruption de changement de niveau est exécutée à partir du vecteur d'interruption PCI1. Les interruptions PCINT7..0 sont activées individuellement

par le registre PCMSK0.

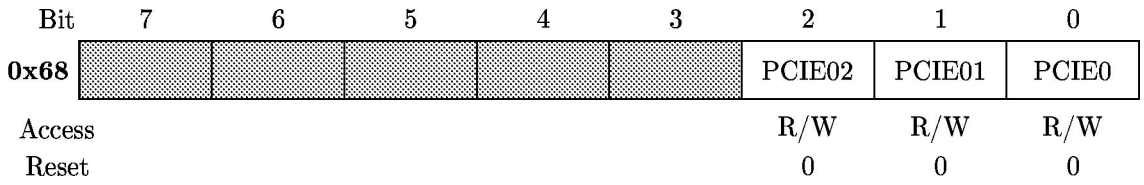


FIGURE 7.8 – Pin Change Interrupt Control Register (PCICR)

Pin Change Interrupt Control Register (PCIFR)

- **Bit 2 - PCIF2: Pin Change Interrupt Flag 2** Lorsqu'un changement de logique sur n'importe quel broche PCINT23..16 déclenche une demande d'interruption, le PCIF2 devient actif. Si le bit I dans SREG et le bit PCIE2 dans PCICR sont au niveau haut, le MCU branchera au vecteur d'interruption correspondant. Le drapeau est effacé lorsque la routine d'interruption est exécutée. Il est également possible d'effacer le drapeau en y inscrivant un niveau logique 1.
- **Bit 1 - PCIF1: Pin Change Interrupt Flag 1** Lorsqu'un changement de logique sur n'importe quel broche PCINT14..8 déclenche une demande d'interruption, le PCIF1 devient actif. Si le bit I dans SREG et le bit PCIE1 dans PCICR sont au niveau haut, le MCU branchera au vecteur d'interruption correspondant. Le drapeau est effacé lorsque la routine d'interruption est exécutée. Il est également possible d'effacer le drapeau en y inscrivant un niveau logique 1.
- **Bit 0 - PCIF0: Pin Change Interrupt Flag 0** Lorsqu'un changement de logique sur n'importe quel broche PCINT7..0 déclenche une demande d'interruption, le PCIF0 devient actif. Si le bit I dans SREG et le bit PCIE0 dans PCICR sont au niveau haut, le MCU branchera au vecteur d'interruption correspondant. Le drapeau est effacé lorsque la routine d'interruption est exécutée. Il est également possible d'effacer le drapeau en y inscrivant un niveau logique 1.

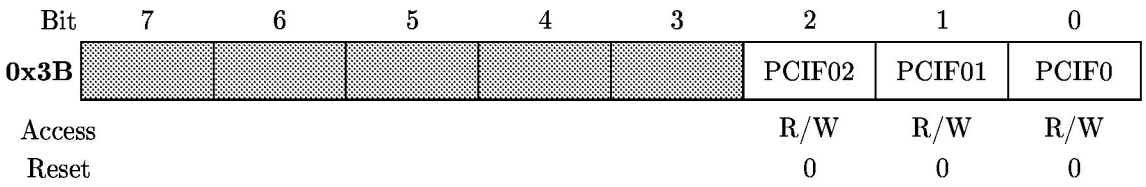


FIGURE 7.9 – Pin Change Interrupt Flag Register (PCIFR)

Pin Change Mask Register 0

- **Bit 7..0 PCINT7..0: Pin Change Enable Mask 7..0**
  Chaque bit PCINT7..0 sélectionne si l'interruption de changement d'état est activée sur la broche d'E/S correspondante. Si l'interruption PCINT7..0 est activée et que le bit PCIE0 dans PCICR est activé, l'interruption de changement de broche est activée sur la broche d'E/S correspondante. Si PCINT7..0 est effacé, l'interruption de changement d'état sur la broche d'E/S correspondante est désactivée.

Bit	7	6	5	4	3	2	1	0
0x6B	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

FIGURE 7.10 – Pin Change Mask Register 0 (PCMSK0)

Pin Change Mask Register 1

- **Bit 14..8 PCINT14..8: Pin Change Enable Mask 14..8**
  Chaque bit PCINT14..8 sélectionne si l'interruption de changement d'état est activée sur la broche d'E/S correspondante. Si l'interruption PCINT14..8 est activée et que le bit PCIE1 dans PCICR est activé, l'interruption de changement de broche est activée sur la broche d'E/S correspondante. Si PCINT14..8 est effacé, l'interruption de changement d'état sur la broche d'E/S correspondante est désactivée.

Bit	7	6	5	4	3	2	1	0
0x6C		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

FIGURE 7.11 – Pin Change Mask Register 1 (PCMSK1)

Pin Change Mask Register 2

- **Bit 23..16 PCINT23..16: Pin Change Enable Mask 23..16**
  Chaque bit PCINT23..16 sélectionne si l'interruption de changement d'état est activée sur la broche d'E/S correspondante. Si l'interruption PCINT23..16 est activée et que le bit PCIE2 dans PCICR est activé, l'interruption de changement de broche

est activée sur la broche d'E/S correspondante. Si PCINT23..16 est effacé, l'interruption de changement d'état sur la broche d'E/S correspondante est désactivée.

Bit	7	6	5	4	3	2	1	0
0x6D	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCIN17	PCINT16
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

FIGURE 7.12 – Pin Change Mask Register 2 (PCMSK2)

7.13 SCRST - System Control and Reset

7.14 Initialisation du microcontrôleur - SCRST

7.14.1 Initialisation du microcontrôleur

L'initialisation ou la réinitialisation d'un microcontrôleur est essentielle pour son bon fonctionnement, car elle garantit que ses états internes auront des valeurs initiales appropriées. Pendant l'initialisation, tous les registres d'entrée/sortie sont mis à leurs valeurs initiales, et le programme commence l'exécution à partir du Reset Vector. L'instruction placée dans le vecteur Reset doit être une instruction Absolute Jump(JMP), une instruction de saut vers la routine de traitement de l'initialisation. Si le programme n'active pas toujours les sources d'interruption, les vecteurs d'interruption ne sont pas utilisés, et le code peut être placé dans ces positions mémoires.

7.14.2 Les sources de Reset

Le microcontrôleur ATmega328 peut être initialisé par quatre sources. La figure ?? montre la structure du circuit de réinitialisation. Elle montre comment les différentes causes peuvent produire le signal appelé Internal Reset, qui affecte le CPU du microcontrôleur. Les source de l'initialisation sont :

- Mise sous tension: Le microcontrôleur est mis en reset lorsque la tension d'alimentation est inférieure à Power-on Reset threshold (V_{POT}) qui a une valeur typique de 2.7 V.
- Reset externe: Le MCU est initialisé quand un niveau bas est appliqué sur la broche RESET pendant plus de 1.5 micro-seconde, ce qui correspond à la durée minimale requise (t_{RST}).

- Watchdog System Reset: Le microcontrôleur est initialisé lorsque la période Watchdog Timer expire et le mode Watchdog Système Reset est activé.
- Brown-out Reset: Le microcontrôleur est initialisé lorsque le détecteur Brown-Out est activé et la tension d'alimentation V_{CC} est inférieure au seuil fixé Brown-Out Reset threshold(V_{BOT}). La valeur du V_{BOT} est configurable à 2.7 V ou 4.0 V, et le temps minimum nécessaire (t_{BOT}) pour considérer une réduction de tension est de 2 micro-secondes.

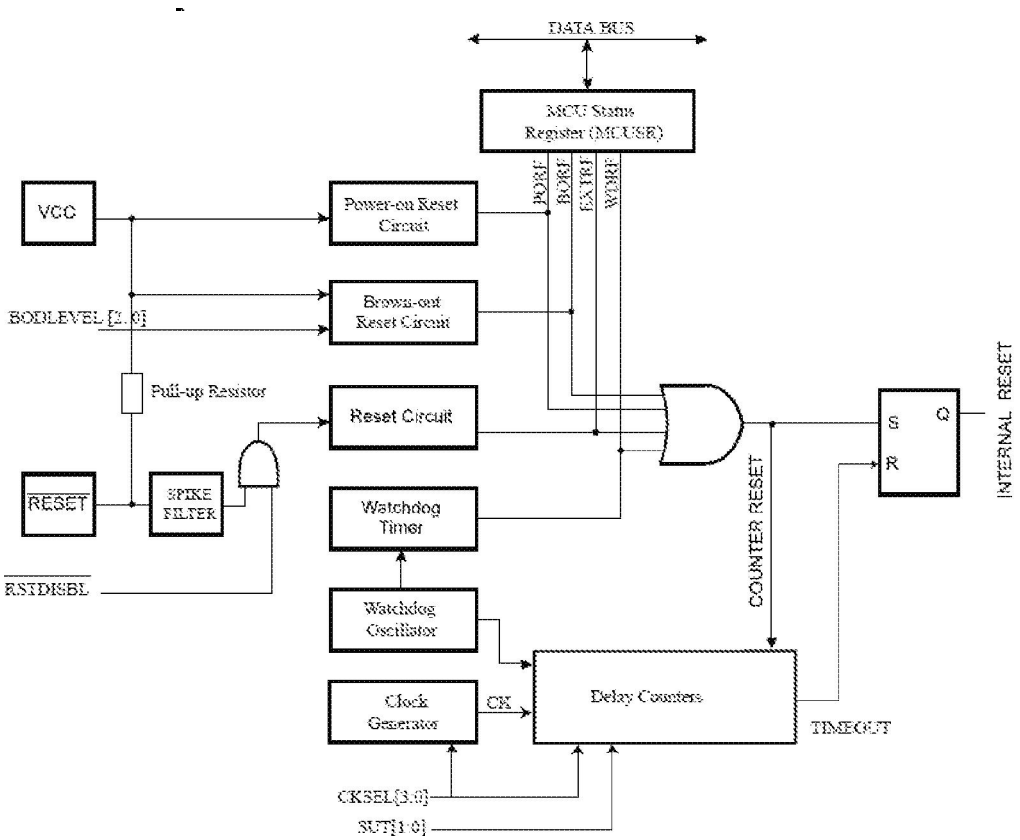


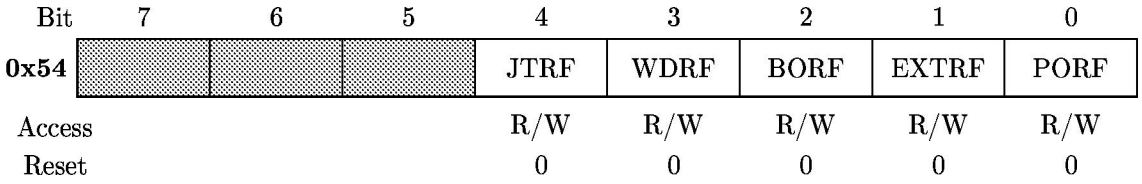
FIGURE 7.13 – Logique de Reset du microcontrôleur Atmaega328p

Comme il existe différentes causes de réinitialisation, le microcontrôleur comprend un registre de contrôle et d'état de l'unité centrale (MCUSR, MCU State Register) dans lequel la cause de réinitialisation est indiquée par un drapeau. Les bits du registre du MCUSR sont:

- **Bit 3 - WDRF Watchdog System Reset Flag:** Ce bit est activé si une réinitialisation Watchdog System se produit. Le bit est remis à zéro par une réinitialisation à la mise sous tension, ou en y inscrivant un 0.
- **Bit 2 BORF - Brown-out Reset Flag**: Ce bit est activé si une réinitialisation

Brown-out se produit. Le bit est remis à zéro par une réinitialisation à la mise sous tension, ou en y inscrivant un 0.

- **Bit 1 EXTRF: External Reset Flag:** Ce bit est activé si une réinitialisation externe se produit. Le bit est remis à zéro par une réinitialisation à la mise sous tension, ou en y inscrivant un 0.
- **Bit 0 - PORF Power-on Reset Flag** Ce bit est activé si une réinitialisation de mise sous tension se produit. Le bit est remis à zéro par une réinitialisation à la mise sous tension, ou en y inscrivant un 0.



Bit	7	6	5	4	3	2	1	0
0x54				JTRF	WDRF	BORF	EXTRF	PORF
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

FIGURE 7.14 – MCU Control and State Register(MCUSR)

7.14.3 Les interruptions des timers

7.15 Les interruptions dans la carte Arduino

Il existe 26 sources d'interruption pour le microcontrôleur Armega328. Dans cette section, nous allons nous concentrer seulement sur celles que nous pouvons utiliser dans les sketchs Arduino. Il s'agit des 5 interruptions liées à l'état ou au changement de l'état de la tension présente sur une broche numérique qui sont donc ce que l'on appelle des interruptions externes, ces interruptions sont représentées dans le tableau **??**. Les interruptions internes liées au timers, des compteurs internes à l'Arduino qui permettent de compter le temps de manière précise ne sont pas étudiées dans cette section.

Source	Rôle
INT0	Interruption externe sur la broche 2
INT1	Interruption externe sur la broche 3
PCINT0	Interruption externe sur la broche 8 à 13
PCINT1	Interruption externe sur la broche A0 à A5
PCINT3	Interruption externe sur la broche A0 à A5

TABLE 7.4 – Les interruptions externes de l'Arduino

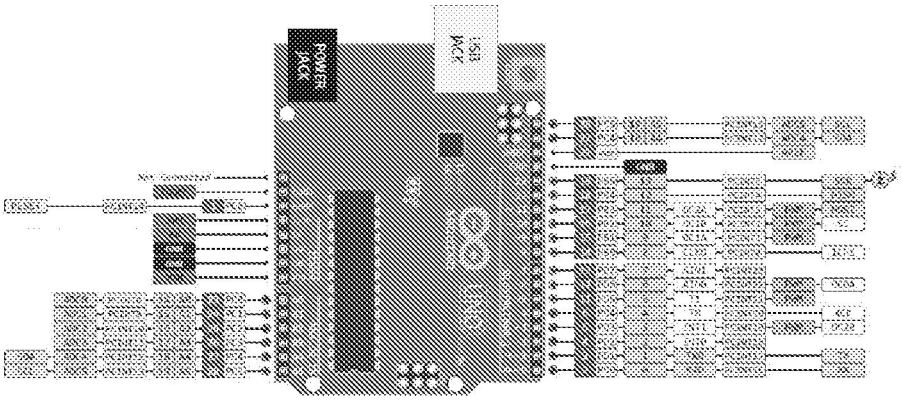


FIGURE 7.15 – Brochage de la carte Arduino Uno

7.15.1 Les interruptions externes INT0 et INT1

Ces deux interruptions sont les plus faciles à utiliser. Pour la carte Arduino Uno à base de l'Atmega328, INT0 correspond à la broche 2 et INT1 correspond à la broche 3.

Appel direct de la fonction d'Arduino

Il est nécessaire de lier la source de l'interruption INT0 ou INT1 avec la routine d'interruption à exécuter lorsque l'interruption est déclenchée. On utilise la fonction `attachInterrupt()` qui prend 3 arguments: le numéro de l'interruption externe, la fonction à appeler quand l'interruption survient et enfin la condition selon laquelle l'interruption est déclenchée. Son prototype est le suivant:

```
attachInterrupt(numero, ISR, mode);
```

Les arguments de la fonction sont:

- `numero` : est le numéro de l'interruption concernée. Il s'agira de 0 ou 1 sur un Arduino Uno, ce qui correspond respectivement aux broches 2 et 3. Sur un Arduino Mega, les numéros 0 à 5 seront possibles et correspondent, dans l'ordre, aux broches 21, 20, 19, 18, 2 et 3.
- `ISR` ; est la routine d'interruption qui sera appelée lorsque l'interruption surviendra. Cette routine d'interruption est une fonction qui ne renvoie rien et qui ne prend aucun argument.
- `mode` : indique ce qui va provoquer l'interruption. Les valeurs possibles pour mode sont:
  - **LOW :** l'interruption est déclenchée quand la broche concernée est LOW. Comme il s'agit d'un état et non d'un événement, l'interruption sera déclenchée tant que la broche est LOW. Par conséquent, dès que l'ISR aura terminée son

exécution, elle la recommencera. Pendant ce temps, loop() ne sera pas exécuté.

- **CHANGE :** l'interruption est déclenchée quand la broche concernée change d'état, c'est à dire passe de LOW à HIGH ou bien de HIGH à LOW. Il s'agit d'un événement.
- **RISING**: l'interruption est déclenchée quand la broche concernée passe de LOW à HIGH. Il s'agit également d'un événement.
- **FALLING**: l'interruption est déclenchée quand la broche concernée passe de HIGH à LOW. Il s'agit encore d'un événement.

Les modes de déclenchement sont le reflet direct des capacités du matériel et à ce titre permettent la meilleure réactivité. Il faut noter tout de même qu'il s'écoule presque 3.5626 micro-secondes entre le déclenchement de l'interruption et l'exécution de la première instruction de l'ISR.

Une deuxième fonction, `detachInterrupt()` permet de déconnecter l'ISR de la source d'interruption. Son prototype est le suivant :

```
detachInterrupt(numero);
```

Où `numero` est le numéro d'interruption, 0 ou 1 pour un Arduino Uno, 0 à 5 pour un Arduino Mega.

Les fonctions `interrupts()` et `noInterrupts()` permettent respectivement d'activer ou de désactiver les interruptions. Celles-ci sont activées par défaut.

Deux broches d'interruptions externes ne permettent pas de mettre en œuvre des systèmes de grande taille. Heureusement, toutes les broches numériques de l'Arduino Uno peuvent servir comme broches d'interruptions externes.

Attention, à l'intérieur de la fonction attachée à l'interruption, certaines fonctions sont inutilisables; la fonction delay ne fonctionne pas et la valeur renvoyée par millis ne s'incrémente pas. Les données séries reçues pendant l'exécution de la fonction sont perdues et les signaux PWM sont affectés. Toute valeur modifiée à l'intérieur de la routine d'interruption devra être déclarée comme volatile, afin que le processeur aille chercher la valeur en mémoire et ne se fie pas à ce qui se trouve dans ses registres qui étaient gelés au moment de l'interruption.

Si une fonction `attachInterrupts` assigne une routine à une broche, celle-ci écrase la configuration précédente de la broche. La fonction `detachInterrupt(pin)` permet de retirer l'assignation d'une interruption à une pin.

En utilisant les registres internes du microcontrôleur

Comme indiqué précédemment, en plus du registre d'état, trois registre sont utilisés pour la gestion de ces deux interruptions.

- Le bit 7 du registre SREG permet d'autoriser/interdire les interruptions. Ce bit

est mis à 1 par l'appel de la fonction `sei()` et mis à zéro pal la fonction `cli()`. On peut aussi, modifier sa valeur par une écriture directe dans le registre SREG: Bit 7 - I Global Interrupt Enable.

- Ensuite il est nécessaire de valider l'interruption en mettant à 1 le bit INTx correspondant dans le registre EIMSK.
- Pour choisir le mode de déclenchement de l'interruption, on configure le registre EICRA (de la même façon pour INT0 et INT1).
- Enfin il est nécessaire de lier la routine d'interruption avec la source d'interruption. Pour cela on utilise la fonction ISR de la façon suivante:

```
ISR(INT0_vect) {
  ......  // Routine d'IT associ\'{e}e \`{a} l'interruption INT0
 }
```

7.15.2 Les interruptions externes PCINT0, PCINT1 et PCINT2

Ces interruptions externes offrent moins de possibilités que celles que nous avons vues au paragraphe précédent. Tout d'abord, les 3 sources PCINT0, PCINT1 et PCINT2 sont partagées par plusieurs broches. PCINT0 correspond aux interruptions externes sur les broches 8 à 13, PCINT1 aux interruptions externes sur les broches A0 à A5 et PCINT3 aux interruptions externes sur les broches 0 à 7. On peut noter que les broches 2 et 3 sont donc partagées entre PCINT3 et INT0 et INT1. On ne peut donc à la fois bénéficier de INT0 et INT1 et des interruptions PCINT3 sur les broches 2 et 3. Comme la même ISR est exécutée pour toutes les broches qui partagent sa source d'interruption, la discrimination de la broche qui a engendré l'interruption, ou des broches en cas de changements simultanés, doit être faite dans la routine d'interruption. Ensuite, la richesse des modes de déclenchement de INT0 et INT1 n'existent pas pour les sources PCINTx. Le seul mode est l'interruption sur changement d'état, LOW vers HIGH et HIGH vers LOW. Par conséquent si seul l'un des changements d'état intéresse le programme, il faut également discriminer en allant lire l'état de la broche qui a déclenché l'interruption.

Heureusement il existe une bibliothèque qui gère toutes ces situations et qui permet, d'une part, de retrouver une souplesse et une richesse comparable à celle de INT0 et INT1 et, d'autre part, de masquer la complexité du matériel.

Il faut toutefois garder à l'esprit que, comparé aux interruptions INT0 et INT1, l'usage des interruptions PCINTx conduira à une exécution plus fréquente de l'ISR et à une réactivité moindre puisque la bibliothèque prendra du temps pour déterminer la broche source et le type de changement avant d'appeler la routine d'interruption. On privilégiera donc les premières. Le corollaire est que la fonction à utilisée n'est pas directement une

ISR mais une fonction appelée par l'ISR. En effet, il n'y a que 3 ISR, une pour chacune des trois sources PCINT0, PCINT1 et PCINT2, mais on peut définir autant de fonctions qu'il y a de broches grâce à la bibliothèque.

7.16 Exemples

Dans cette section, des exemples d'utilisation des interruptions externes sont présentés, documentant les aspects de la programmation qui doivent être pris en compte lors du développement d'autres applications. 7.13.1 Exemple 1: 7.13.2 Exemple 2: Supposons qu'on veut mesurer la vitesse de rotation d'un moteur avec une carte Arduino en utilisant un encodeur. Si on veut que le programme compte le nombre des impulsions venant de l'encodeur tournant, sans jamais négliger une impulsion, il serait très compliqué d'écrire un programme pour faire quoique ce soit, parce que le programme devrait en permanence examiner les broches connectées à l'encodeur, afin de prendre en compte les impulsions lorsqu'elle surviendront. Dans une situation pareille, l'utilisation des interruptions libère le microcontrôleur pour faire d'autres choses tant que l'évènement attendu ne survient pas.

```
const int MOTEUR = 9; // Digital pin pour commande moteur
int val,V;
unsigned int tick_codeuse = 0; // Compteur de tick de la codeuse
/* Routine d'initialisation */
void setup() {
Serial.begin(9600);
pinMode(MOTEUR, OUTPUT); // Sortie moteur
analogWrite(MOTEUR, 0); // Sortie moteur \‘{a} 0
delay(5000); // Pause de 5 sec pour laisser le temps
// au moteur de s'arr\^{e}ter si celui-ci est en marche
attachInterrupt(0, compteur, CHANGE); // Interruption sur tick de la codeuse (interru
}
/* Fonction principale */
void loop(){
val=analogRead(A0);
V = map(val, 0, 1023, 0, 255);
analogWrite(MOTEUR, V);
delay(1000);
Serial.print(" Vitesse =");
Serial.print(tick_codeuse); // Le nombre de Tours Chaque seconde
```

```
Serial.println(" tr/s");
tick_codeuse=0;
}
/* Interruption sur tick de la codeuse */
void compteur(){
tick_codeuse++; // On incr\'{e}mente le nombre de tick de la codeuse
}
```