ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

LES INTERRUPTIONS

Pierre-Yves Rochat

rév 2020/03/25

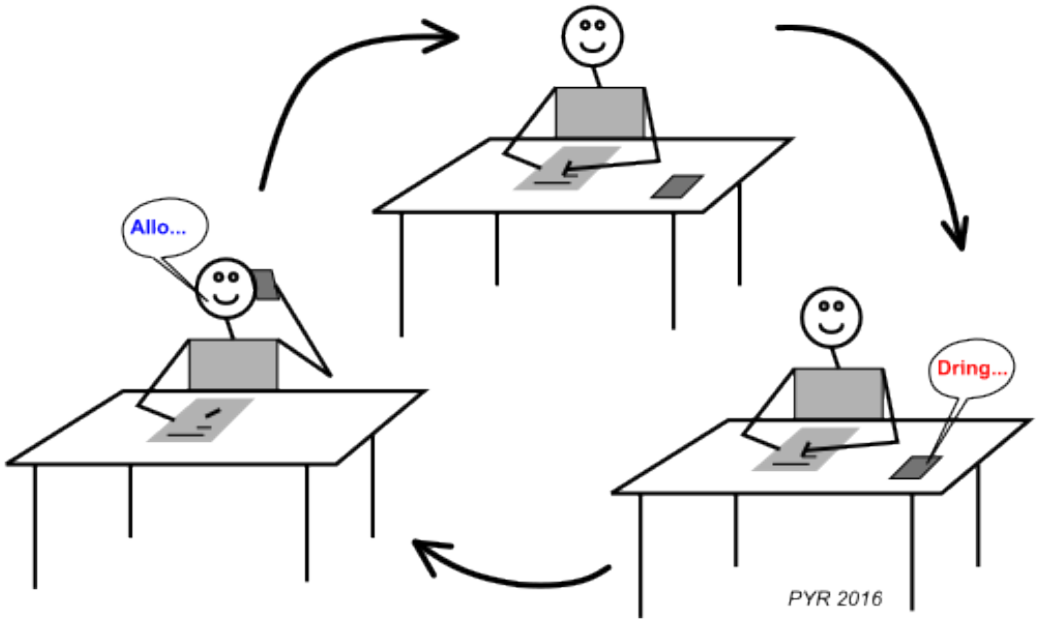
Introduction

Un système à microcontrôleur est généralement pourvu d'entrées et de sorties. Le but premier du programme est souvent de réagir correctement aux changements d'état des entrées, en agissant en conséquence sur les sorties.

La méthode de scrutation impose beaucoup de contraintes temporelles. Les interruptions sont un mécanisme très puissant et très utilisé pour résoudre ce problème.

Définition

On appelle **interruption** dans un système informatique l'arrêt temporaire d'un programme au profit d'un autre programme, jugé à cet instant plus important. L'interruption correspond au sens qu'on donne à ce mot dans nos vies courantes. Prenons un exemple : je suis en train de travailler à mon bureau. Le téléphone sonne. Je vais répondre au téléphone. Après la conversation, je reprends mon travail là où je l'avais laissé.



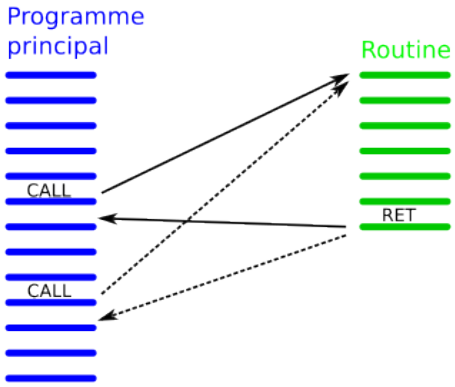
Interruption dans la vie courante

C'est toujours un *événement* qui va produire une interruption. Cet événement a un caractère imprévisible, le programme ne sais pas quand il va se produire.

Implémentation

Pour utiliser les interruptions, il n'est pas indispensable de comprendre en détail le mécanisme qui les rend possibles. C'est comme les procédures ou les fonctions, que nous avons l'habitude d'utiliser sans forcément connaître les mécanismes matériels qui les rendent possibles.

Toutefois, nous allons ici faire une petite incursion dans le monde de la programmation en assembleur, pour mieux comprendre les interruptions. La figure ci-dessous montre un programme, dont les instructions successives sont notées en bleu. Dans le cas où une fraction du programme doit s'exécuter plusieurs fois, on a l'habitude de grouper ses instructions, notées ici en vert. On appelle ce morceau de programme une *routine* ou *sous-routine*. Ce concept correspond aux procédures et aux fonctions dans les langages évolués comme le C.

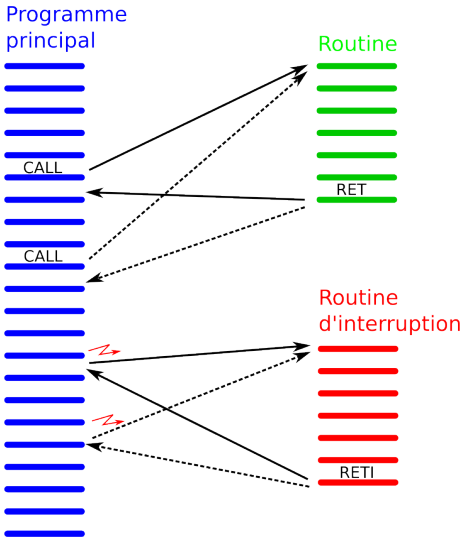


Appel d'une routine

L'appel de la routine se fait par une instruction **Call** dans le programme principal. A la fin de l'exécution de la routine, une instruction **Ret** (return = retour) permet de revenir au programme appelant, juste après l'instruction *Call*. La routine peut être appelée plusieurs fois dans le programme principal.

Notons que l'adresse de retour doit être mémorisée pour que le retour soit possible. C'est une **pile** (*stack*) en mémoire vive qui est utilisée à cette fin. Nous ne détaillerons pas son mécanisme ici.

Regardons maintenant la figure suivante. Un nouvelle routine, appelée **Routine d'interruption** est représentée en rouge. On voit qu'elle va aussi s'exécuter. Mais son exécution n'est pas la conséquence d'une instruction *Call*.



Principe des interruptions

Rien dans le programme principal ne permet de savoir que cette routine va s'exécuter. C'est un **événement** qui est la cause de son exécution. C'est ce qu'on appelle un interruption.

La routine d'interruption se termine aussi par une instruction de retour, appelée **Reti** (*return from interrupt*). En plus d'effectuer le retour au programme interrompu, elle rétablit le mode de sensibilité aux interruptions qui prévalait avant l'interruption.

Nature des événements

Quels sont ces événements qui vont produire une interruption ? Il en existe principalement deux sortes :

- Les événements **extérieurs** au microcontrôleur. Il s'agit par exemple d'un changement sur une entrée.
- Les événements **intérieurs** au microcontrôleur. Par exemple, beaucoup de microcontrôleurs sont pourvus d'un convertisseur analogique-numérique (*ADC = Analog to Digital Converter*). La conversion est déclenchée par un fanion et dure un certain temps. Plutôt que d'attendre la fin de la conversion, le programme principal peut continuer, puis être interrompu au moment de la fin de la conversion.

Dans cette catégorie des interruptions intérieures au microcontrôleur, les plus importantes sont celle liées aux **Timers**. Ce sujet sera abordé dans un chapitre séparé, vu son importance pour la commande des enseignes et afficheurs à LED.

Discrimination des sources d'interruption

Il existe généralement plusieurs sources interruptions sur un microcontrôleur. Lorsqu'une interruption se produit, le système doit être capable d'en savoir la source. Si rien n'est prévu au niveau matériel, la routine d'interruption doit consulter les registres pour chaque interruption, pour connaître celle qui a été activée.

Les **vecteurs d'interruption** (*interrupt vectors*) permettent d'être plus efficace : une adresse différente est réservée pour le début de la routine de chaque interruption.

Souvent ces deux mécanismes vont être utilisés successivement, comme par exemple lors d'une interruption produite par une entrée sur un MSP430.

A titre d'exemple, voici la table résumée des vecteurs d'interruption pour un MSP430G, y compris l'adresse pour le Reset :

- 0xFFFE : Reset
- 0xFFFC : NMI
- 0xFFFA : Timer1 CCR0
- 0xFFF8 : Timer1 CCR1, CCR2, TAIFG
- 0xFFF6 : Comparator_A
- 0xFFF4 : Watchdog Timer
- 0xFFF2 : Timer0 CCR0
- 0xFFF0 : Timer0 CCR1, CCR2, TAIFG
- 0xFFEE : USCI status
- 0xFFEC : USCI receive/transmit
- 0xFFEA : ADC10
- 0xFFE8 : -
- 0xFFE6 : Port P2
- 0xFFE4 : Port P1

Les adresses se trouvent en mémoire flash, ce sont les dernières adresses de l'espace d'adressage de 16 bits.

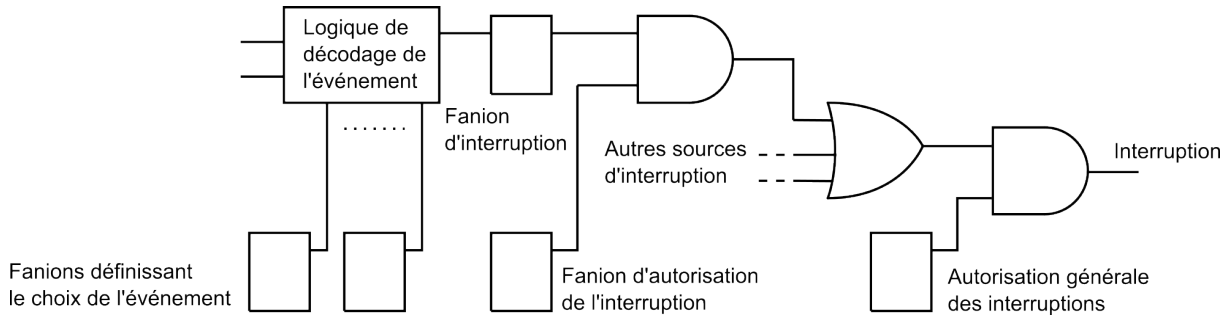
Plusieurs sources d'interruptions nécessitent la scrutation pour déterminer la cause exacte de l'interruption. C'est le cas par exemple des interruptions sur les ports : chaque bit peut produire une interruption. C'est aussi le cas d'une des interruptions des Timers : les registre de comparaison 1 et 2, ainsi que l'interruption générale du Timer sont regroupés sur un vecteur unique.

Mise en œuvre d'une interruption

Plusieurs étapes sont nécessaire pour mettre en œuvre une routine d'interruption :

1. Enclencher l'interruption qui nous intéresse. Par exemple une interruption sur une entrée.
2. Préciser comment cette interruption doit fonctionner. Par exemple dire sur quel flanc l'interruption doit se produire.
3. Enclencher globalement les interruptions. Les microprocesseurs disposent d'un fanion général qui autorise les interruptions. Il est souvent utilisé pour pouvoir éviter les interruptions dans certaines parties critiques au programme, qui ne doivent pas être interrompues.

Le schéma logique ci-dessous montre la logique qui permet de générer les interruptions et les fanions qu'il faut ajuster.



Logique de génération des interruptions

On y trouve :

- la logique qui permet de saisir un événement
- les fanions qui règlent la manière dont l'événement est décodé
- le fanion qui enclenche cette interruption particulière
- la porte ET associée à ce fanion pour produire cette interruption
- la porte OU qui permet à toutes les interruptions d'être prises en compte
- le fanion général d'autorisation des interruptions
- la porte ET qui produit finalement les interruptions.