CPE LYON

LIVE AND DISCOVER

Bases des systèmes embarqués

A01- Mise en place des Interruptions dans le 8051F020

Version : 2022

09/11/2022 12:05

UNIVERSITÉ DE LYON
établissement associé

Bases des systèmes embarqués

A01- Mise en place des Interruptions dans le 8051F020

Définitions – Présentation des mécanismes de gestion d'interruption

Mise en place des interruptions

Au fait, c’est quoi une interruption?

Une interruption: pour quoi faire?

Définition: Une interruption désigne un mécanisme à la fois matériel et logiciel qui permet de suspendre temporairement l'exécution d'un programme pour exécuter un programme prioritaire chargé de traiter l'arrivée d'un évènement extérieur au processeur

- Les interruptions sont utilisées pour gérer des évènements non prévisibles externes **au processeur** (Central Processing Unit)
- Elles permettent d'éviter au processeur de scruter en permanence l'arrivée d'un évènement.
- Elles permettent au processeur d'exécuter de manière quasi immédiate du code pour traiter un évènement externe

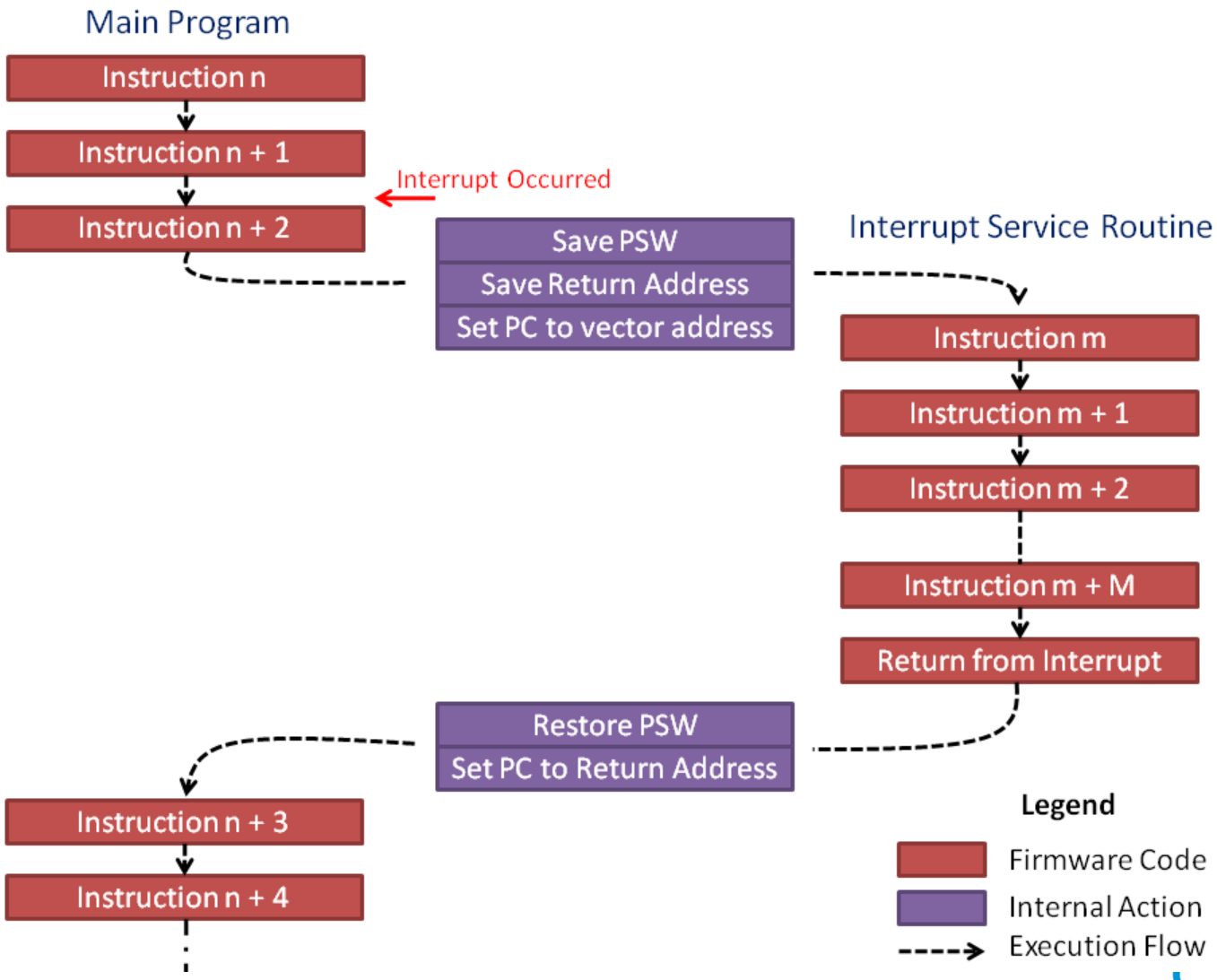
Attention, on parle ici d'évènement externes au processeur. Aussi dans un microcontrôleur les périphériques intégrés génèrent des évènement externes au processeur, et peuvent donc produire des interruptions.

Mise en place d'une interruption

Quel que soit le processeur, la mise en place d'une interruption requiert 2 éléments:

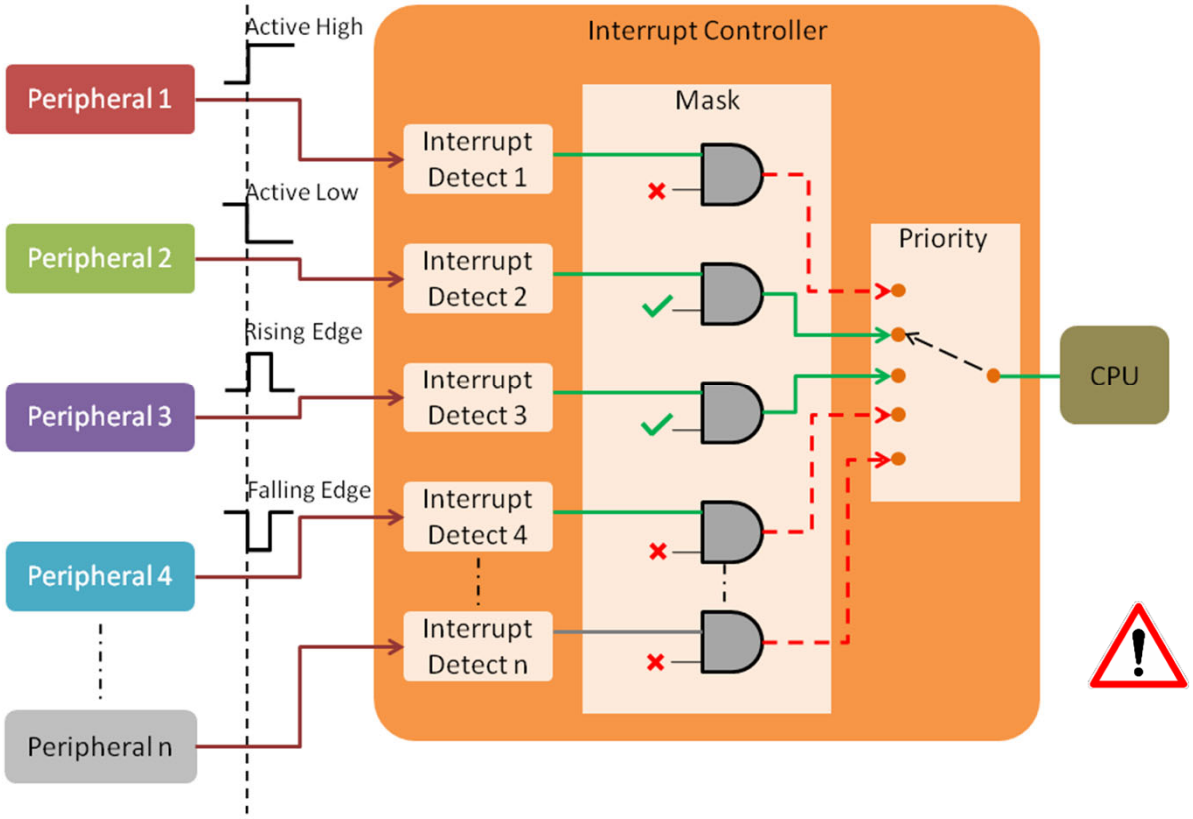
- Elément 1 - Un code d'interruption et son vecteur associé
- Elément 2 - Une configuration de cette interruption (autorisation, priorisation, conditions de déclenchement….)

Mécanisme d'interruption « classique » au niveau assembleur



Source: http://www.embien.com/blog/interrupt-handling-in-embedded-software/

Validation et priorisation des interruptions



1. Les interruptions peuvent en général être masquées (autorisées ou pas)
2. Les interruptions sont classées par priorité (2 niveaux sur le 8051F020)
3. Selon les priorités définies, une interruption peut être interruptible. Une interruption de priorité basse pourra être interrompue par une interruption de priorité plus haute

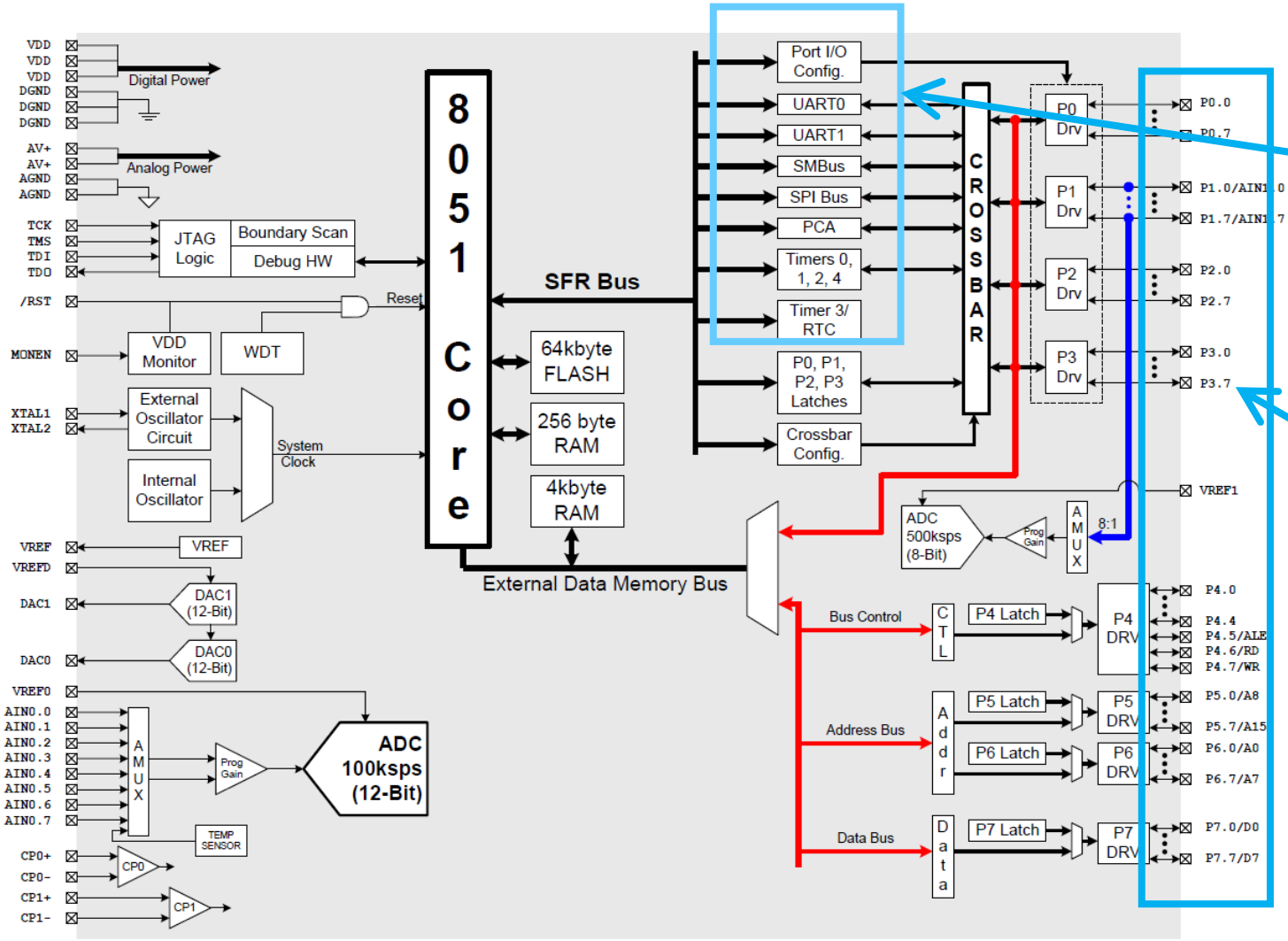
Règle de base: en général, un programme d'interruption se doit d'être le plus court possible.

Formulé autrement: il est impératif de bien maitriser le temps d'exécution d'un programme d'interruption

IMPORTANT

Source: http://www.embien.com/blog/interrupt-handling-in-embedded-software/

Sources d'interruptions dans le 8051F020

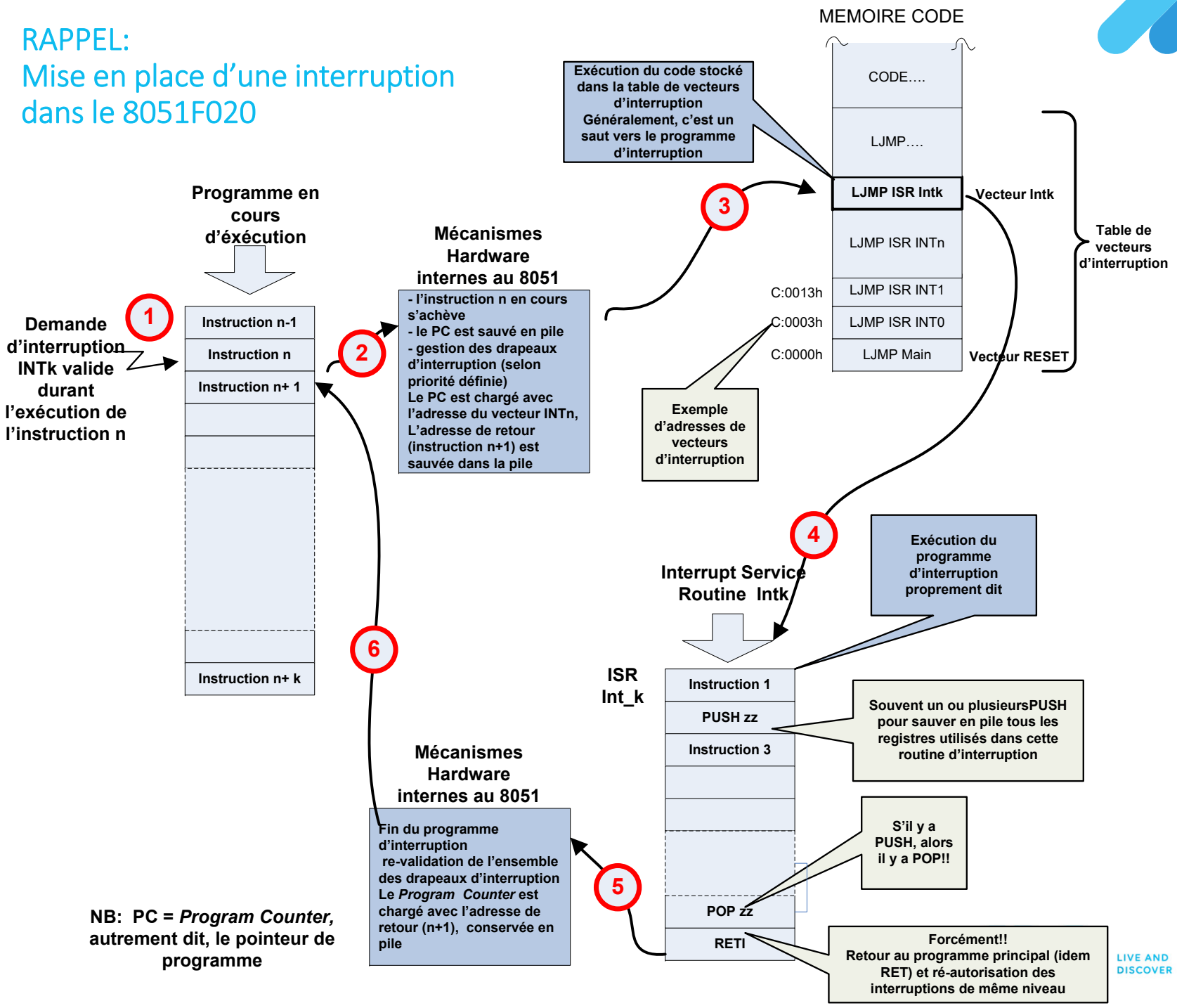


Sources d'interruptions internes: produites par les périphériques internes

IMPORTANT

Sources d'interruptions externes: produites par des signaux extérieurs connectés sur des entrées spécifiques d'interruption

RAPPEL:
Mise en place d'une interruption dans le 8051F020



NB: PC = *Program Counter*, autrement dit, le pointeur de programme

Programmation des interruptions dans un 8051

Programmer (autoriser, prioriser, configurer…) une interruption revient à lire et écrire dans des registres dédiés à la gestion des diverses interruptions

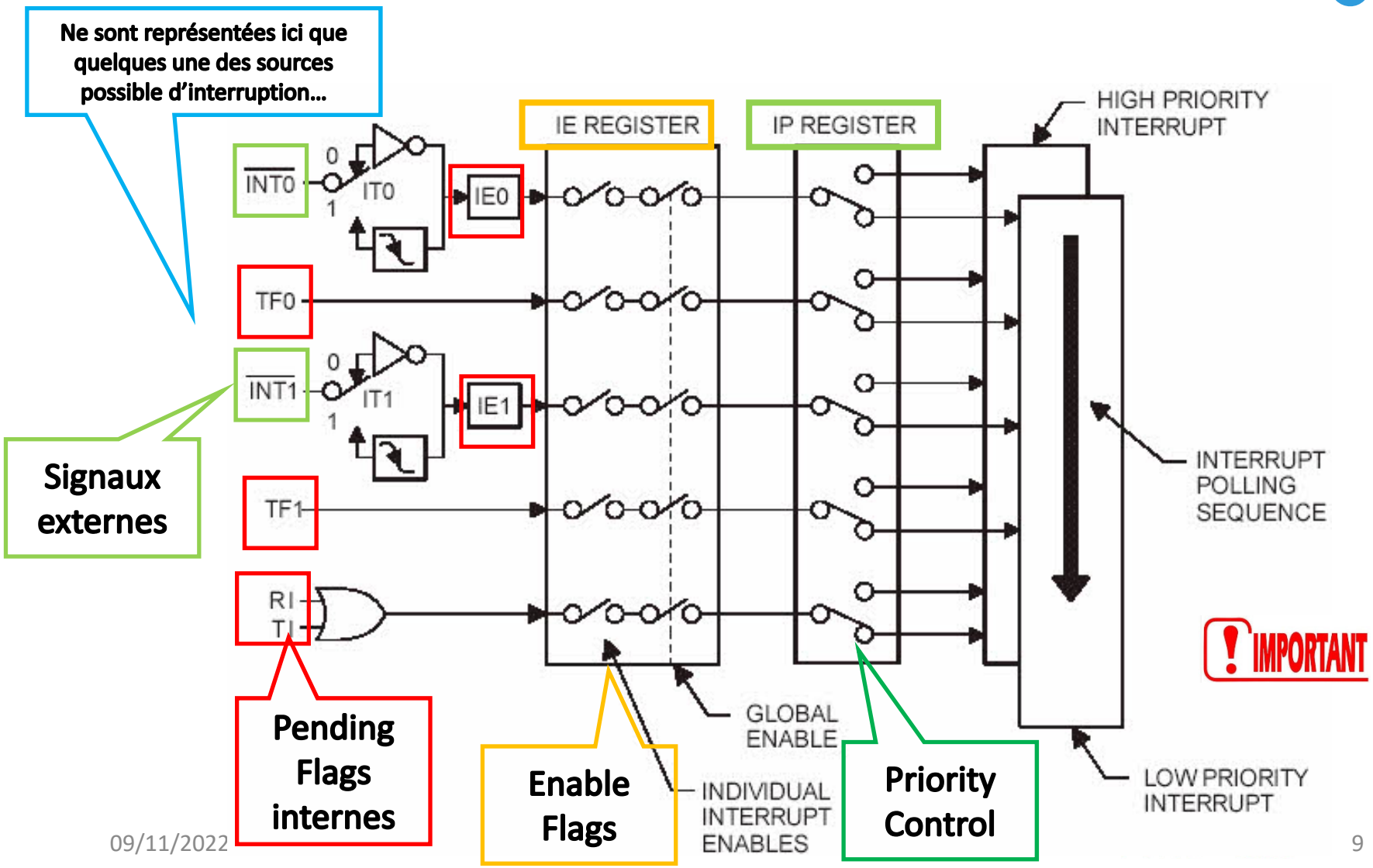


Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow (or EXF2)	0x002B	5	TF2 (T2CON.7)	Y		ET2 (IE.5)	PT2 (IP.5)
Serial Peripheral Interface	0x0033	6	SPIF (SPI0CN.7)	Y		ESPI0 (EIE1.0)	PSPI0 (EIP1.0)
SMBus Interface	0x003B	7	SI (SMB0CN.3)	Y		ESMB0 (EIE1.1)	PSMB0 (EIP1.1)
ADC0 Window Comparator	0x0043	8	AD0WINT (ADC0CN.2)	Y		EWADC0 (EIE1.2)	PWADC0 (EIP1.2)
Programmable Counter Array	0x004B	9	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y		EPCA0 (EIE1.3)	PPCA0 (EIP1.3)
Comparator 0 Falling Edge	0x0053	10	CP0FIF (CPT0CN.4)			ECP0F (EIE1.4)	PCP0F (EIP1.4)
Comparator 0 Rising Edge	0x005B	11	CP0RIF (CPT0CN.5)			ECP0R (EIE1.5)	PCP0R (EIP1.5)
Comparator 1 Falling Edge	0x0063	12	CP1FIF (CPT1CN.4)			ECP1F (EIE1.6)	PCP1F (EIP1.6)
Comparator 1 Rising Edge	0x006B	13	CP1RIF (CPT1CN.5)			ECP1R (EIE1.7)	PCP1F (EIP1.7)
Timer 3 Overflow	0x0073	14	TF3 (TMR3CN.7)			ET3 (EIE2.0)	PT3 (EIP2.0)
ADC0 End of Conversion	0x007B	15	AD0INT (ADC0CN.5)	Y		EADC0 (EIE2.1)	PADC0 (EIP2.1)
Timer 4 Overflow	0x0083	16	TF4 (T4CON.7)			ET4 (EIE2.2)	PT4 (EIP2.2)
ADC1 End of Conversion	0x008B	17	AD1INT (ADC1CN.5)			EADC1 (EIE2.3)	PADC1 (EIP2.3)
External Interrupt 6	0x0093	18	IE6 (P3IF.5)			EX6 (EIE2.4)	PX6 (EIP2.4)
External Interrupt 7	0x009B	19	IE7 (P3IF.6)			EX7 (EIE2.5)	PX7 (EIP2.5)
UART1	0x00A3	20	RI1 (SCON1.0) TI1 (SCON1.1)			ES1	PS1
External Crystal OSC Ready	0x00AB	21	XTLVLD (OSCXCN.7)			EXVLD (EIE2.7)	PXVLD (EIP2.7)

Table des vecteurs d'interruption dans le 8051F020

Eléments sur lesquels on va pouvoir agir:

- Drapeaux (Pending Flag) dans les **Registres de périphériques**
- Autorisation (Enable flag) **IE + EIE1 + EIE2**
- Priorité (Priority Control) **IP + EIP1 + EIP2**

Définitions:

Pending Flag (Drapeau d'attente?): il signale l'arrivée d'un évènement dans un périphérique ou sur une broche d'interruption **susceptible** de provoquer une interruption

Enable Flag (Drapeau d'autorisation), il autorise la transmission de la demande d'interruption au mécanisme matériel de gestion d'interruption.

Priority Control: fixe la priorité de cette interruption: prioritaire ou pas.

IMPORTANT

Rappel: Procédure de mise en place d'une interruption en **assembleur**

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**

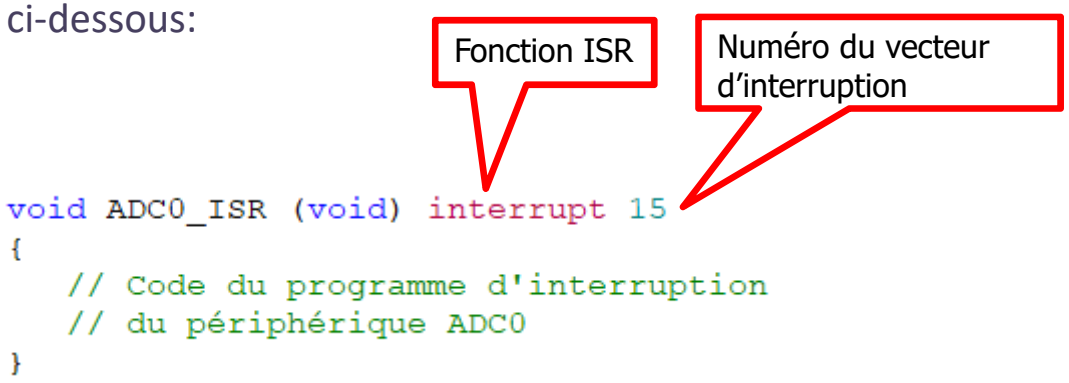
Elément 1 - En C, lors de la déclaration d'une fonction, c'est le mot clé « Interrupt » suivi du numéro de vecteur, qui permet au compilateur de gérer ces 2 étapes

3. Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité.
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

Elément 2 - En C, les opérations à mener sont exactement les mêmes qu'en assembleur, seul le langage diffère

Mise en place d'une « Interrupt Service Routine (ISR) » en C Keil

Les étapes précédentes 1 et 2 sont faites lors de la déclaration de la fonction d'interruption comme ci-dessous:



Les étapes 3 à 6 restent à coder…

Remarque 1: C'est le mot clé « **Interrupt** » qui informe le compilateur que cette fonction devra être traitée comme une fonction d'interruption. Le compilateur va donc devoir se charger de compléter la table de vecteurs d'interruption

Remarque 2: le numéro du vecteur d'interruption, permet au compilateur de déterminer l'adresse à partir de laquelle sera placé le code de saut vers la fonction d'interruption. Ce numéro de vecteur d'interruption est donné dans la documentation constructeur (Table des interruptions, Colonne « Priority order »)

Remarque 3: Le nom de la fonction peut être quelconque, mais l'usage veut que son nom indique clairement qu'il s'agit d'une fonction d'interruption.

IMPORTANT

Un exemple pour comprendre ce que fait le compilateur en présence d'une fonction déclarée comme fonction d'interruption:

Soit la fonction d'interruption suivante:

```
void ADC0_ISR (void) interrupt 15
{
   // Code du programme d'interruption
   // du périphérique ADC0
{
```

A la compilation, le compilateur détecte que la fonction ADC0_ISR doit être traitée comme une fonction d'interruption (grâce au mot clé « Interrupt »).

Timer 3 Overflow	0x0073	14	TF3 (TMR3CN.7)			ET3 (EIE2.0)	PT3 (EIP2.0)
ADC0 End of Conversion	0x007B	15	AD0INT (ADC0CN.5)	Y		EADC0 (EIE2.1)	PADC0 (EIP2.1)
Timer 4 Overflow	0x0083	16	TF4 (T4CON.7)			ET4 (EIE2.2)	PT4 (EIP2.2)

Grâce au numéro de vecteur d'interruption (15) dans la table de vecteurs d'interruptions, le compilateur détermine qu'il doit placer à l'adresse 0X007B, un saut vers le sous-programme (la fonction) ADC0_ISR.

Enfin, le compilateur fait en sorte de terminer le sous-programme ADC0_ISR par l'instruction assembleur RETI au lieu d'un RET

Interruptions dans le 8051F020

Priority Order: Numéro du vecteur d'interruption – utilisé par le compilateur pour déterminer l'adresse du vecteur d'interruption
Il fixe aussi un ordre de priorité

Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		ES0 (IE.4)	PS0 (IP.4)

Priorités « Priority Order » et « Priority Control »

Les interruptions dans les processeurs de la famille 8051 peuvent avoir **2 niveaux** de priorité: Haut et Bas. Ces niveaux de priorité sont fixés pour chaque interruption en agissant sur le **bit « Priority Control »** de chaque source d'interruption.

Les règles de priorité pour ces priorités hautes et basses sont les suivantes:

- Une interruption de priorité Haute en cours d'exécution ne peut pas être interrompue (sauf par un Reset).
- Une interruption de priorité Basse, en cours d'exécution, peut être interrompue par une interruption de priorité haute. A l'issue de l'exécution du code de cette interruption de priorité Haute, l'interruption de priorité Basse pourra s'achever

Par ailleurs, l'ordre d'exécution des interruptions de même niveau de priorité (Priority control) est fixé par la valeur de la colonne « Priory Order». Plus la valeur est petite et plus l'interruption est prioritaire.

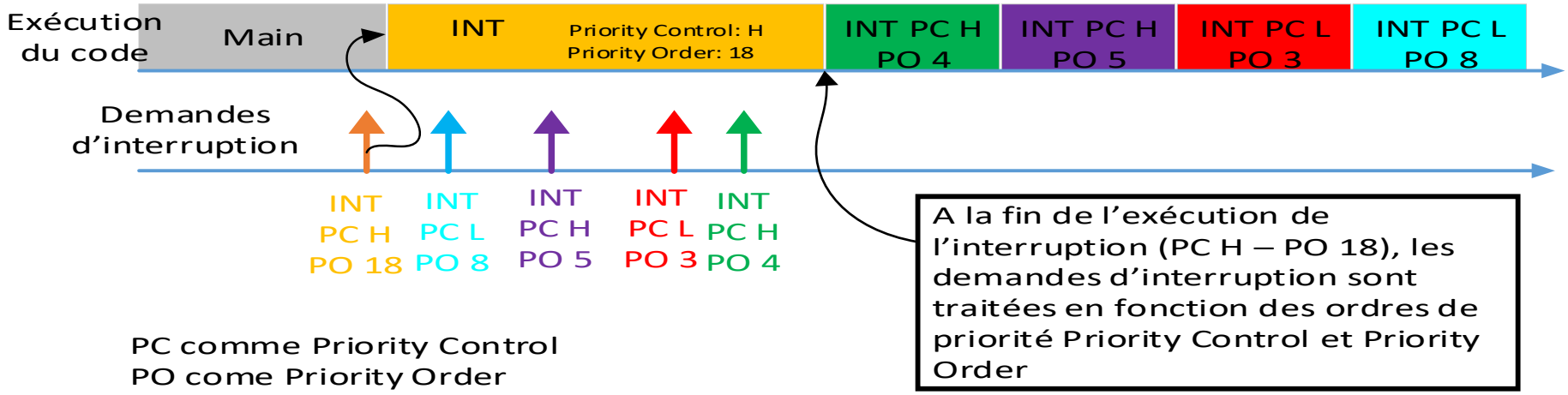


Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	it addressable?	leared by HW?	Enable Flag	Priority Control

Interruptions internes et interruptions externes dans le 8051F020

- Rappel: une source d'interruption est dite **« interne »** lorsqu'elle est produite par un périphérique interne au microcontrôleur: par exemple les timers (0,1…4), les interfaces série UART, SPI, I2C et les convertisseurs analogiques/numériques peuvent générer des demandes d'interruption
- Rappel: une source d'interruption est dite **« externe »** lorsqu'elle est produite par un dispositif externe branché sur une broche du microcontrôleur, broche susceptible (par construction et configuration) de provoquer une demande d'interruption.
- Interruptions externes disponibles sur le 8051F020:
  - INT0 accessible sur une broche P0.0 à P2.3 (selon configuration de la matrice d'interconnexion Crossbar)
  - INT1 accessible sur une broche P0.0 à P2.5 (selon configuration de la matrice d'interconnexion Crossbar)
  - INT6 accessible sur la broche P3.6
  - INT7 accessible sur la broche P3.7

Utilisation de la documentation: Informations relatives aux registres

Chaque registre est décrit en détail dans un tableau récapitulant de nombreuses informations le concernant

Figure 22.5. TCON: Timer Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0x88

Bit7: TF1: Timer 1 Overflow Flag.
Set by hardware when Timer 1 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine.
0: No Timer 1 overflow detected.
1: Timer 1 has overflowed.

Bit6: TR1: Timer 1 Run Control.
0: Timer 1 disabled.
1: Timer 1 enabled.

Bit5: TF0: Timer 0 Overflow Flag.
Set by hardware when Timer 0 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine.
0: No Timer 0 overflow detected.
1: Timer 0 has overflowed.

Accessibilité des bits en lecture ou en écriture

Indique si le registre est adressable bit à bit

Contenu du registre après Reset

Description du rôle de chaque bit

Rappel: gestion des interruptions dans un 8051

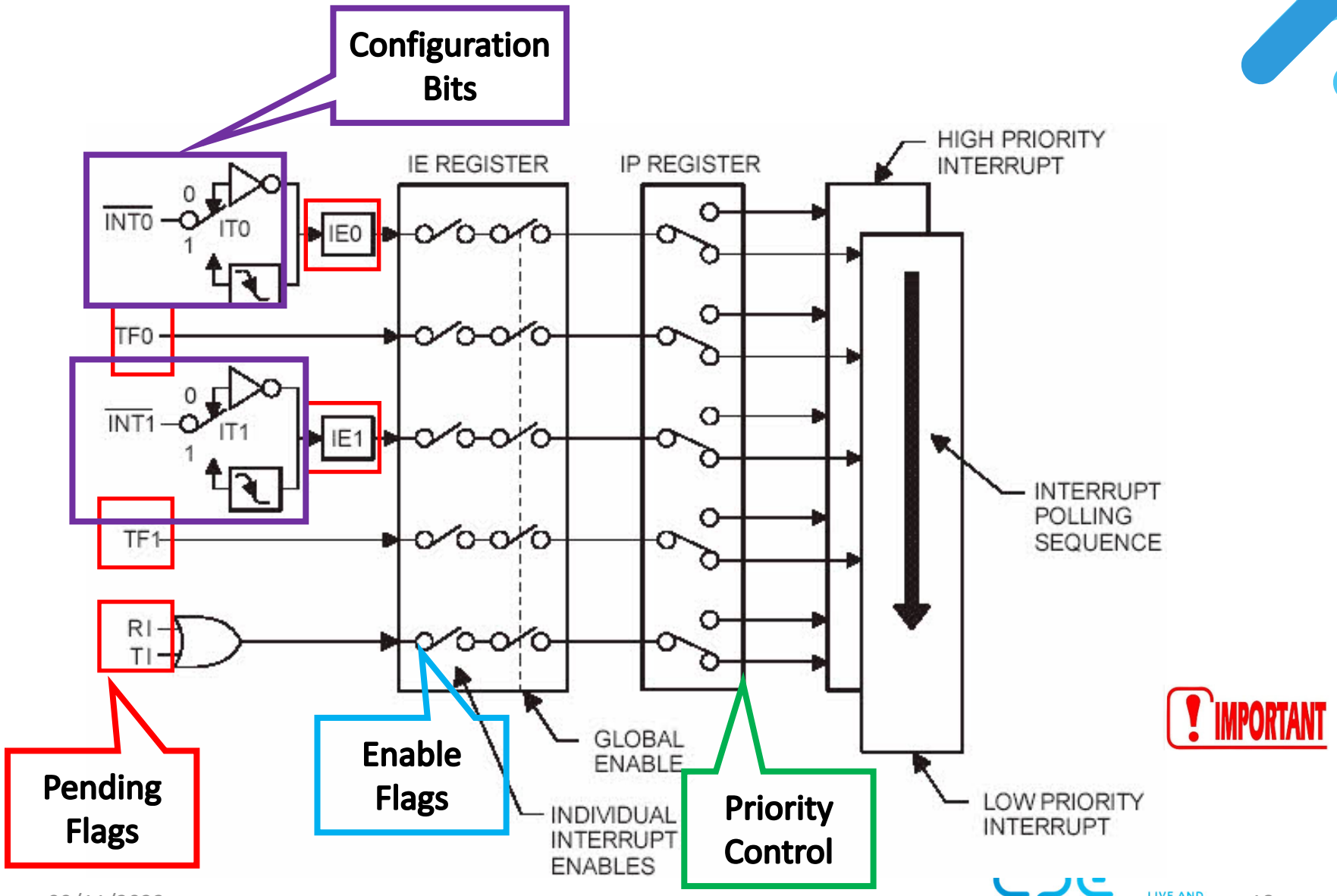


Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		ES0 (IE.4	PS0 (IP.4)
Timer 2 Overflow (or EXF2)	0x002B	5	TF2 (T2CON.7)	Y		ET2 (IE.5	PT2 (IP.5)
Serial Peripheral Interface	0x0033	6	SPIF (SPI0CN.7)	Y		ESPI0 (EIE1.0)	PSPI0 (EIP1.0)
SMBus Interface	0x003B	7	SI (SMB0CN.3)	Y		ESMB0 (EIE1.1)	PSMB0 (EIP1.1)
ADC0 Window Comparator	0x0043	8	AD0WINT (ADC0CN.2)	Y		EWADC0 (EIE1.2)	PWADC0 (EIP1.2)
Programmable Counter Array	0x004B	9	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y		EPCA0 (EIE1.3)	PPCA0 (EIP1.3)
Comparator 0 Falling Edge	0x0053	10	CP0FIF (CPT0CN.4)			ECP0F (EIE1.4)	PCP0F (EIP1.4)
Comparator 0 Rising Edge	0x005B	11	CP0RIF (CPT0CN.5)			ECP0R (EIE1.5)	PCP0R (EIP1.5)
Comparator 1 Falling Edge	0x0063	12	CP1FIF (CPT1CN.4)			ECP1F (EIE1.6)	PCP1F (EIP1.6)
Comparator 1 Rising Edge	0x006B	13	CP1RIF (CPT1CN.5)			ECP1R (EIE1.7)	PCP1F (EIP1.7)
Timer 3 Overflow	0x0073	14	TF3 (TMR3CN.7)			ET3 (EIE2.0)	PT3 (EIP2.0)
ADC0 End of Conversion	0x007B	15	AD0INT (ADC0CN.5)	Y		EADC0 (EIE2.1)	PADC0 (EIP2.1)
Timer 4 Overflow	0x0083	16	TF4 (T4CON.7)			ET4 (EIE2.2)	PT4 (EIP2.2)
ADC1 End of Conversion	0x008B	17	AD1INT (ADC1CN.5)			EADC1 (EIE2.3)	PADC1 (EIP2.3)
External Interrupt 6	0x0093	18	IE6 (P3IF.5)			EX6 (EIE2.4)	PX6 (EIP2.4)
External Interrupt 7	0x009B	19	IE7 (P3IF.6)			EX7 (EIE2.5)	PX7 (EIP2.5)
UART1	0x00A3	20	RI1 (SCON1.0) TI1 (SCON1.1)			ES1	PS1
External Crystal OSC Ready	0x00AB	21	XTLVLD (OSCXCN.7)			EXVLD (EIE2.7)	PXVLD (EIP2.7)

Table des vecteurs d'interruption dans le 8051F020

Eléments sur lesquels on va pouvoir agir:

- Drapeaux (Pending Flag) dans les **Registres de périphériques**
- Autorisation IE + **EIE1 + EIE2**
- Priorité IP + **EIP1 + EIP2**

Définitions:

Pending Flag (Drapeau d'attente?): il signale l'arrivée d'un évènement dans un périphérique ou sur une broche d'interruption **susceptible** de provoquer une interruption

Enable Flag (Drapeau d'autorisation), il autorise la transmission de la demande d'interruption au mécanisme matériel de gestion d'interruption.

Priority Control: fixe la priorité de cette interruption: prioritaire ou pas.

IMPORTANT

EXEMPLE Interruption INT7: Registre d'état, de validation, de priorité et de configuration

Registre EIE2

Figure 12.12. EIE2: Extended Interrupt Enable 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EXVLD	ES1	EX7	EX6	EADC1	ET4	EADC0	ET3	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE7

Bit7: EXVLD: Enable External Clock Source Valid (XTLVLD) Interrupt.
This bit sets the masking of the XTLVLD interrupt.
0: Disable XTLVLD interrupt.
1: Enable interrupt requests generated by the XTLVLD flag (OSCXCN.7)

Bit6: ES1: Enable UART1 Interrupt.
This bit sets the masking of the UART1 interrupt.
0: Disable UART1 interrupt.
1: Enable UART1 interrupt.

Bit5: EX7: Enable External Interrupt 7.
This bit sets the masking of External Interrupt 7.
0: Disable External Interrupt 7.
1: Enable interrupt requests generated by the External Interrupt 7 input pin.

Bit4: EX6: Enable External Interrupt 6.
This bit sets the masking of External Interrupt 6.
0: Disable External Interrupt 6.
1: Enable interrupt requests generated by the External Interrupt 6 input pin.

Registre EIP2

Figure 12.14. EIP2: Extended Interrupt Priority 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PXVLD	EP1	PX7	PX6	PADC1	PT4	PADC0	PT3	
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

Bit7: PXVLD: External Clock Source Valid (XTLVLD) Interrupt Priority Control.
This bit sets the priority of the XTLVLD interrupt.
0: XTLVLD interrupt set to low priority level.
1: XTLVLD interrupt set to high priority level.

Bit6: EP1: UART1 Interrupt Priority Control.
This bit sets the priority of the UART1 interrupt.
0: UART1 interrupt set to low priority.
1: UART1 interrupt set to high priority.

Bit5: PX7: External Interrupt 7 Priority Control.
This bit sets the priority of the External Interrupt 7.
0: External Interrupt 7 set to low priority level.
1: External Interrupt 7 set to high priority level.

Bit4: PX6: External Interrupt 6 Priority Control.
This bit sets the priority of the External Interrupt 6.
0: External Interrupt 6 set to low priority level.
1: External Interrupt 6 set to high priority level.

Figure 17.19. P3IF: Port3 Interrupt Flag Register

Registre P3IF

R/W	R/W	R	R	R/W	R/W	R/W	R/W	Reset Value
IE7	IE6	-	-	IE7CF	IE6CF	-	-	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xAD

Bit7: IE7: External Interrupt 7 Pending Flag
0: No falling edge has been detected on P3.7 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.7 is detected.

Bit6: IE6: External Interrupt 6 Pending Flag
0: No falling edge has been detected on P3.6 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.6 is detected.

Bits5-4: UNUSED. Read = 00b, Write = don't care.

Bit3: IE7CF: External Interrupt 7 Edge Configuration
0: External Interrupt 7 triggered by a falling edge on the IE7 input.
1: External Interrupt 7 triggered by a rising edge on the IE7 input.

Bit2: IE6CF: External Interrupt 6 Edge Configuration
0: External Interrupt 6 triggered by a falling edge on the IE6 input.
1: External Interrupt 6 triggered by a rising edge on the IE6 input.

Bits1-0: UNUSED. Read = 00b, Write = don't care.

Exemple de l'interruption externe INT7

Figure 12.9. IE: Interrupt Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	IEGF0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: (bit addressable) 0xA8

Bit7: EA: Enable All Interrupts.
This bit globally enables/disables all interrupts. When set to '0', individual interrupt mask settings are overridden.
0: Disable all interrupt sources.
1: Enable each interrupt according to its individual mask setting.

Bit6: IEGF0: General Purpose Flag 0.

IMPORTANT

Règle de base pour l'utilisation des interruptions

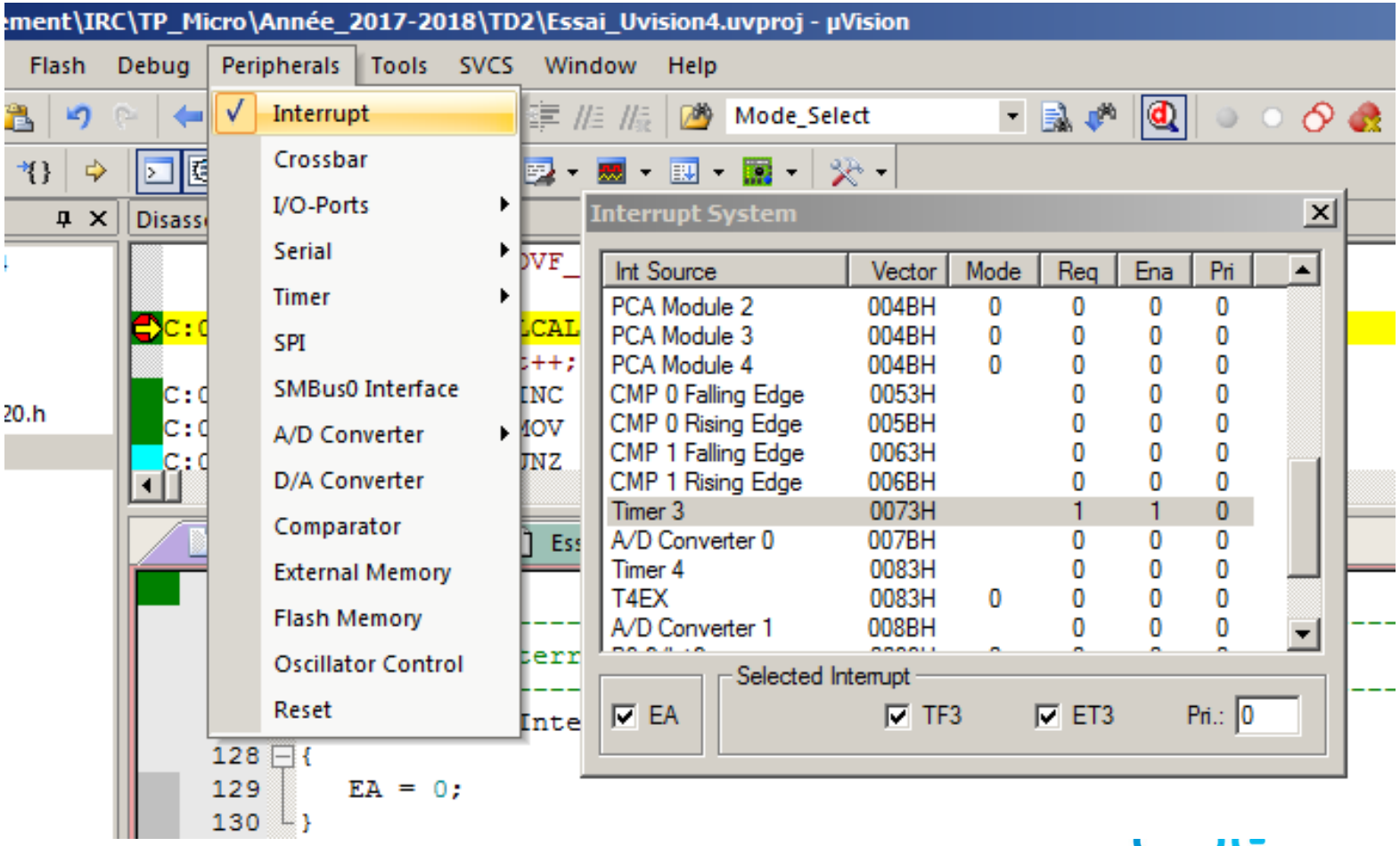
Avant d'autoriser une interruption pour un périphérique donné, on écrit d'abord le programme d'interruption correspondant!

Car, utiliser un périphérique ne veut pas forcément dire que l'on va utiliser l'interruption de ce périphérique

Erreur classique: une interruption autorisée, mais pas de code d'interruption: plantage garanti!!

Déboggueur et interruptions…

Au cour d'une session de débogage, La fenêtre « Interrupt » permet de savoir à tout instant quelles sont les potentielles interruptions autorisées, quelles sont les demandes en cours, etc…



Bases des systèmes embarqués

A01- Mise en place des Interruptions dans le 8051F020

Mise en œuvre d'un cas pratique – INT7

Cas pratique

Une application logicielle provoque (entre autre) une incrémentation toutes les 10 ms d'un compteur 10 bits (comptage de 0 à 1023). Par ailleurs, les 8 bits de poids fort de ce compteur sont envoyés sur le port 7.

On souhaite à l'aide d'un signal (SIG_IN) externe au microcontrôleur remettre à zéro la sortie le compteur.
Contrainte: la remise à zéro doit être effectuée le plus rapidement possible dès que le signal appelé SIG_IN passe au niveau haut.
La durée du niveau haut de ce signal peut varier de 1µS à 1s.

Solution logicielle par scrutation (polling)

```
unsigned int compteur_logiciel=0;
//Boucle infinie
while(1)
{
 // Application basique
 compteur_logiciel++;
 if (compteur_logiciel >= 1024) compteur_logiciel = 0;
 P7 = compteur_logiciel >>2; // envoi sur P7 des 8 bits de pds fort
 Delay(1); // Fonction logicielle de temporisation 10ms
}
```

Pour la gestion de la remise à zéro (à l'aide de SIG_IN), On fait quoi?

Solution logicielle par scrutation (polling) avec gestion de la remise à zéro SIG_IN

```
unsigned int compteur_logiciel=0;
// Config SIG_IN
…
//Boucle infinie
while(1)
{
 // Application basique
 compteur_logiciel++;
 if ((compteur_logiciel > 1024) || (SIG_IN == 1))
compteur_logiciel = 0;
 P7 = compteur_logiciel >>2;
 Delay(1); // 10ms
}
```

Sous-entendu: on a configuré une broche d'un port en entrée nommée SIG_IN

Avantage et inconvénient de la solution « Scrutation » (polling)

Avantages:

- Simplicité de mise en œuvre

Inconvénients:

1. Temps de réponse variable – Cas où l'évènement survient durant la fonction delay – on détecte donc à 10ms près (car la durée de la fonction delay est de 10ms)
2. Cas où SIG_IN est très long: fonctionnement par niveau (on peut trouver des solutions logicielles pour résoudre ce problème)
3. Cas où SIG_IN est bref: <10ms – On n'est pas sûr de le détecter (impose une scrutation avec une récurrence inférieure à la micro-seconde!!)

Résolution du cas SIG_IN >> T delay

```
unsigned int compteur_logiciel=0;
// Config SIG_IN
…
bit SIG_IN_OLD = 0;
bit SIG_IN_NOW = 0;
// Boucle infinie
while(1)
{   // Application basique
 compteur_logiciel++;
 if ((SIG_IN_OLD == 0)& (SIG_IN == 0)
     {SIG_IN_NOW = 0; SIG_IN_OLD = 0; }    //Pas d'impulsion SIG_IN
 elsif ((SIG_IN_OLD == 0)& (SIG_IN == 1))   //Nouvelle impulsion H
      {SIG_IN_NOW = 1; SIG_IN_OLD = 1; }
 elsif ((SIG_IN_OLD == 1)& (SIG_IN == 0)) // FIN impulsion H
       {SIG_IN_NOW = 0;  SIG_IN_OLD = 0; }
 elsif ((SIG_IN_OLD == 1)& (SIG_IN == 1)) // impulsion H en cours
       {SIG_IN_NOW = 0; SIG_IN_OLD = 1; }
 if ((compteur_logiciel > 1024) || (SIG_IN_NOW == 1))
       { compteur_logiciel = 0; }
 P7 = compteur_logiciel >>2;
 Delay(1); // 10ms
}
```

Sous-entendu: on a configuré une broche d'un port en entrée nommé SIG_IN

Une solution « Hardware » - Détecteur de front actif

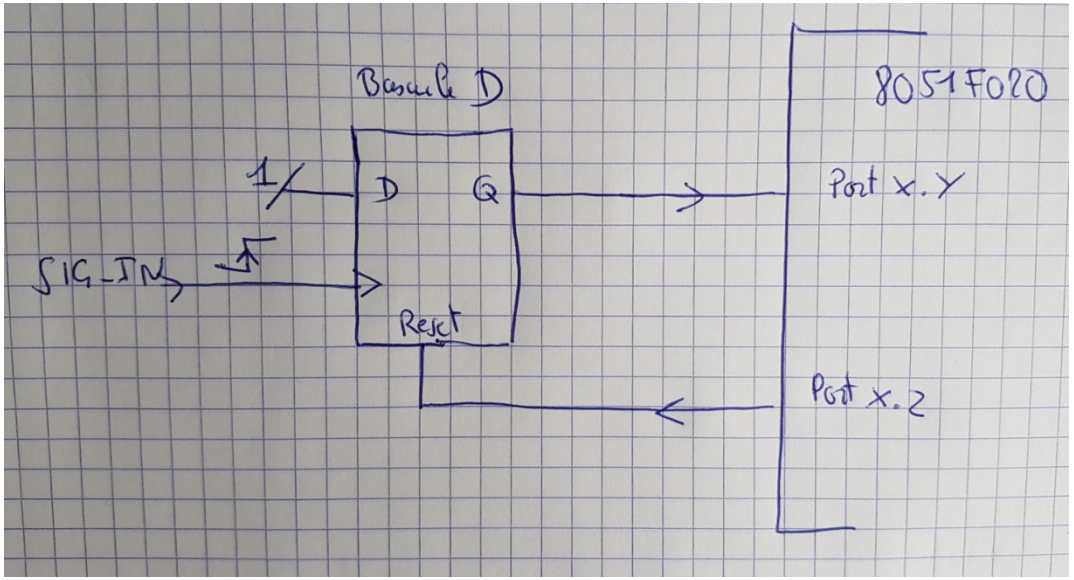
Objectif: résoudre le point suivant: faire en sorte que la détection de SIG_IN fonctionne quelle que soit la durée de SIG_IN

Solution: Une bascule D (sensible sur front montant) et 2 ports (1 entrée – 1 sortie)

Cette solution résout les 2 problèmes: Sig_IN très court (qq US, donc << période scrutation) ou très long (>> période scrutation)

La bascule D mémorise le front montant de Sig_IN, jusqu'à la scrutation

Après scrutation, la bascule est remise à zéro



Implémentation logicielle de la solution Hardware

```
unsigned int compteur_logiciel=0;
// Config SIG_IN
…
// Config RAZ_BasculeD
 …
// Boucle infinie
while(1)
{
 // Application basique
 compteur_logiciel++;
 if ((compteur_logiciel > 1024) || (SIG_IN == 1))
{        compteur_logiciel = 0;
         RAZ_BasculeD = 1;
         RAZ_BasculeD = 0;
}
 P7 = compteur_logiciel >>2;
 Delay(1); // 10ms
}
```

Sous-entendu: on a configuré un port en entrée nommé SIG_IN et un port en sortie RAZ_BasculeD

Avantage et inconvénient de la solution Hardware

Avantages:
Fait le job en résolvant les problèmes soulevés lors de la variation de durée de SIG_IN

Inconvénients:
- On rajoute du hardware: modification de la carte électronique, peu de souplesse.
- Toujours le problème du temps de réponse à l'impulsion (0 < T réponse< durée de la fonction « delay »).

Utiliser des Interruptions?

On souhaite s'affranchir des 2 inconvénients de la solution « Hardware »…

- L'utilisation d'une interruption va permettre de résoudre le problème

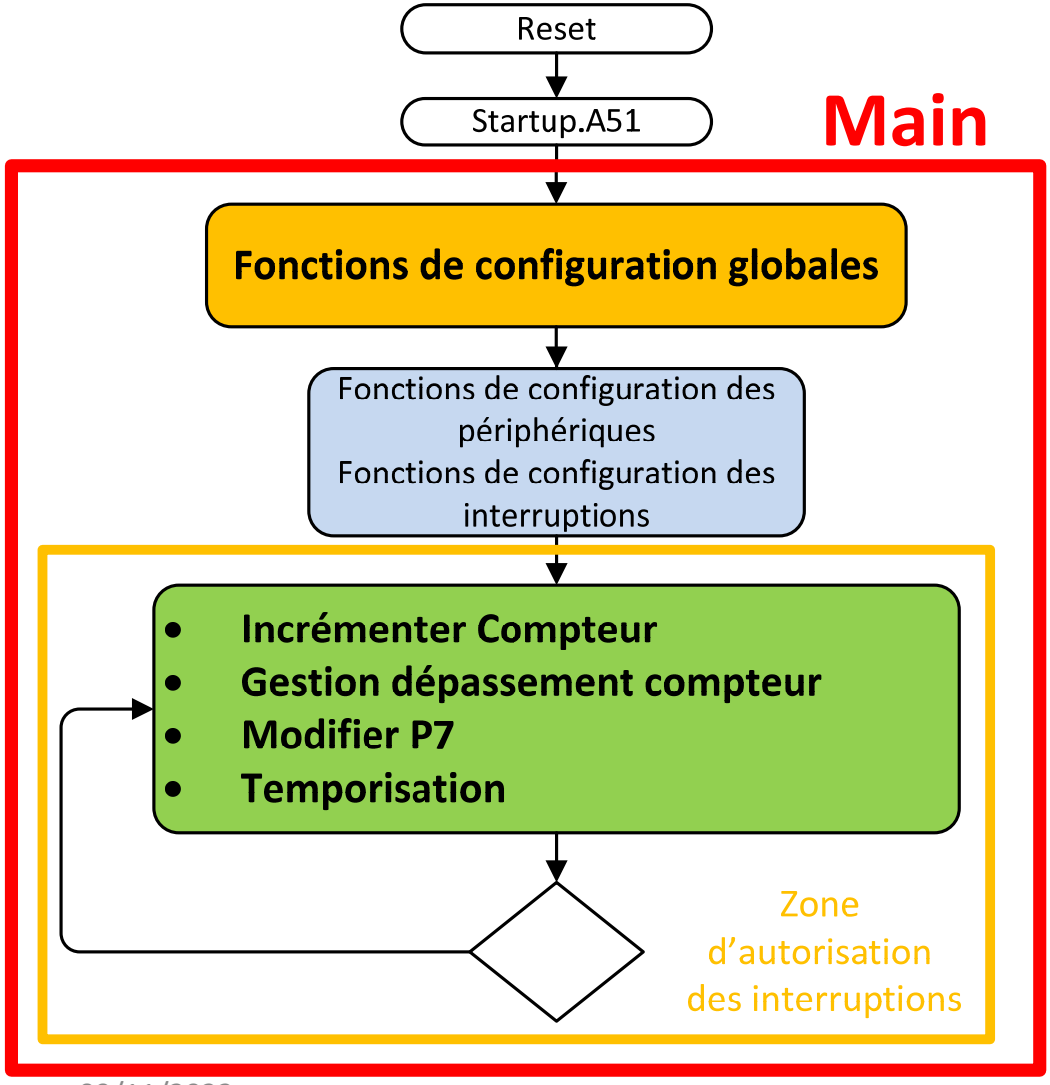
Comment utiliser une interruption? Le signal SIG_IN va être utilisé comme source d'interruption. Ainsi, il sera possible de remettre à zéro le compteur dès un signal actif SIG_IN, quel que soit l'exécution du code en cours/

Quelle Interruption? SIG_IN est extérieur au microcontrôleur, on va donc devoir mettre en œuvre une interruption externe

Mode de déclenchement? Idéalement on souhaiterait que le front montant de SIG_IN provoque une interruption

Quelle configuration?

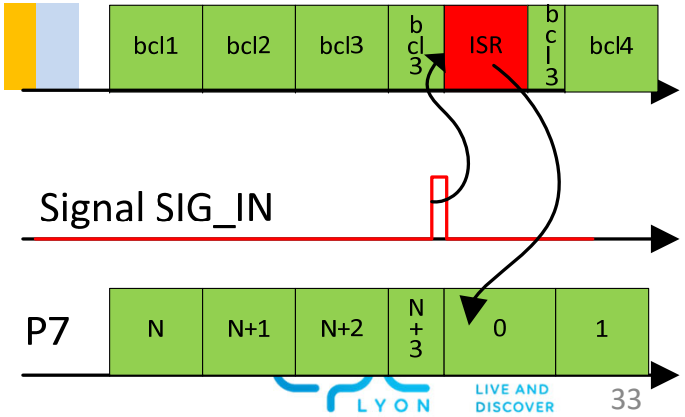
Mode d'utilisation de l'interruption



Interrupt Service Routine
Source de déclenchement:
Front montant sur le signal SIG_IN

ISR
- Remise à zéro Compteur
- Ecriture dans P7

Exécution Code



La solution Interruption avec sources de déclenchement externe

L'objectif est de câbler le signal SIG_IN sur une broche d'interruption du processeur.
On dispose sur le 8051F020 de 4 entrées d'interruption externes:

- INT0, INT1, sensibles sur niveau bas ou sur front descendant (configurable). L'affectation sur les broches de INT0 et INT1 est géré via une matrice d'interconnexion (CROSSBAR).
- INT6 et INT7, sensible sur front montant ou descendant (configurable) Par construction INT6 est reliée à P3.6 et INT7 à P3.7.

Compte tenu du dispositif (impulsion positive sur SIG_IN), on va utiliser une entrée d'interruption sensible sur front montant: INT6 est choisie (INT7 est aussi utilisable)
Résumons: à chaque front montant sur le signal SIG_IN, on va déclencher l'exécution d'un programme d'interruption INT6

INT6 dans la table de vecteur d'interruption

Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
ADC1 End of Conversion	0x008B	17	AD1INT (ADC1CN.5)			EADC1 (EIE2.3)	PADC1 (EIP2.3)
External Interrupt 6	0x0093	18	IE6 (P3IF.5)			EX6 (EIE2.4)	PX6 (EIP2.4)
External Interrupt 7	0x009B	19	IE7 (P3IF.6)			EX7 (EIE2.5)	PX7 (EIP2.5)

EXEMPLE Interruption INT6: Registre d'état, de validation, de priorité et de configuration

Registre EIE2

Figure 12.12. EIE2: Extended Interrupt Enable 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EXVLD	ES1	EX7	EX6	EADC1	ET4	EADC0	ET3	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE7

Bit7: EXVLD: Enable External Clock Source Valid (XTLVLD) Interrupt.
This bit sets the masking of the XTLVLD interrupt.
0: Disable XTLVLD interrupt.
1: Enable interrupt requests generated by the XTLVLD flag (OSCXCN.7)

Bit6: ES1: Enable UART1 Interrupt.
This bit sets the masking of the UART1 interrupt.
0: Disable UART1 interrupt.
1: Enable UART1 interrupt.

Bit5: EX7: Enable External Interrupt 7.
This bit sets the masking of External Interrupt 7.
0: Disable External Interrupt 7.
1: Enable interrupt requests generated by the External Interrupt 7 input pin.

Bit4: EX6: Enable External Interrupt 6.
This bit sets the masking of External Interrupt 6.
0: Disable External Interrupt 6.
1: Enable interrupt requests generated by the External Interrupt 6 input pin.

Registre EIP2

Figure 12.14. EIP2: Extended Interrupt Priority 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
PXVLD	EP1	PX7	PX6	PADC1	PT4	PADC0	PT3	
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	S

Bit7: PXVLD: External Clock Source Valid (XTLVLD) Interrupt Priority Control.
This bit sets the priority of the XTLVLD interrupt.
0: XTLVLD interrupt set to low priority level.
1: XTLVLD interrupt set to high priority level.

Bit6: EP1: UART1 Interrupt Priority Control.
This bit sets the priority of the UART1 interrupt.
0: UART1 interrupt set to low priority.
1: UART1 interrupt set to high priority.

Bit5: PX7: External Interrupt 7 Priority Control.
This bit sets the priority of the External Interrupt 7.
0: External Interrupt 7 set to low priority level.
1: External Interrupt 7 set to high priority level.

Bit4: PX6: External Interrupt 6 Priority Control.
This bit sets the priority of the External Interrupt 6.
0: External Interrupt 6 set to low priority level.
1: External Interrupt 6 set to high priority level.

Registre P3IF

Figure 17.19. P3IF: Port3 Interrupt Flag Register

R/W	R/W	R	R	R/W	R/W	R/W	R/W	Reset Value
IE7	IE6	-	-	IE7CF	IE6CF	-	-	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xAD

Bit7: IE7: External Interrupt 7 Pending Flag
0: No falling edge has been detected on P3.7 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.7 is detected.

Bit6: IE6: External Interrupt 6 Pending Flag
0: No falling edge has been detected on P3.6 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.6 is detected.

Bit5-4: UNUSED. Read = 00b, Write = don't care.

Bit3: IE7CF: External Interrupt 7 Edge Configuration
0: External Interrupt 7 triggered by a falling edge on the IE7 input.
1: External Interrupt 7 triggered by a rising edge on the IE7 input.

Bit2: IE6CF: External Interrupt 6 Edge Configuration
0: External Interrupt 6 triggered by a falling edge on the IE6 input.
1: External Interrupt 6 triggered by a rising edge on the IE6 input.

Bit1-0: UNUSED. Read = 00b, Write = don't care.

Figure 12.9. IE: Interrupt Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	IEGF0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: (bit addressable) 0xA8

Bit7: EA: Enable All Interrupts.
This bit globally enables/disables all interrupts. When set to '0', individual interrupt mask settings are overridden.
0: Disable all interrupt sources.
1: Enable each interrupt according to its individual mask setting.

Bit6: IEGF0: General Purpose Flag 0.

Rappel: Procédure de mise en place d'une interruption – Etape 1

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**

1

En C, c'est le mot clé « Interrupt » suivi du numéro de vecteur, qui permet au compilateur de gérer ces 2 étapes

3. Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité.
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

La solution Interruption avec INT6 – Etape 1

Etape 1 : je crée la fonction d'interruption

```
void ISR_INT6() interrupt 18
{
}
```

C'est ce numéro qui informe le compilateur que cette fonction sera codée comme fonction d'interruption INT6

Le nom de la fonction d'interruption peut être quelconque.
Mais par soucis de clarté, l'usage est d'utiliser le préfixe « ISR » « ***Interrupt Service Routine*** » et un suffixe correspondant au nom de l'interruption (ici « INT6 »)

ADC1 End of Conversion	0x008B	17	AD1INT (ADC1CN.5)			EADC1 (EIE2.3)	PADC1 (EIP2.3)
External Interrupt 6	0x0093	18	IE6 (P3IF.5)			EX6 (EIE2.4)	PX6 (EIP2.4)
External Interrupt 7	0x009B	19	IE7 (P3IF.6)			EX7 (EIE2.5)	PX7 (EIP2.5)

Rappel: Procédure de mise en place d'une interruption – Etape 2

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**
3. Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité. **2**
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

La solution Interruption avec INT6 – Etape 2

Etape 2 : Configuration du périphérique pour être en produire de produire des demandes d'interruption et/ou spécification mode de déclenchement interruption – Action sur le registre P3IF
Au passage, remise à zéro du « pending flag »

```
void Config_INT6_Ext(void)
{
P3IF &= ~(1<<6); // Pending flag IE6 remis à zéro (au cas où…)
P3IF |= (1<<2);  // IE7CF=1 -- Déclenchement INT6 sur front montant
}
```

Figure 17.19. P3IF: Port3 Interrupt Flag Register

R/W	R/W	R	R	R/W	R/W	R/W	R/W	Reset Value
IE7	IE6	-	-	IE7CF	IE6CF	-	-	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xAD

Bit7: IE7: External Interrupt 7 Pending Flag
0: No falling edge has been detected on P3.7 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.7 is detected.
Bit6: IE6: External Interrupt 6 Pending Flag
0: No falling edge has been detected on P3.6 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.6 is detected.
Bits5-4: UNUSED. Read = 00b, Write = don't care.
Bit3: IE7CF: External Interrupt 7 Edge Configuration
0: External Interrupt 7 triggered by a falling edge on the IE7 input.
1: External Interrupt 7 triggered by a rising edge on the IE7 input.
Bit2: IE6CF: External Interrupt 6 Edge Configuration
0: External Interrupt 6 triggered by a falling edge on the IE6 input.
1: External Interrupt 6 triggered by a rising edge on the IE6 input.
Bits1-0: UNUSED. Read = 00b, Write = don't care.

Rappel: Procédure de mise en place d'une interruption – Etape 3

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**
3. Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité.
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.** **3**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

La solution Interruption avec INT6 – Etape 3

Etape 3 : Validation interruption INT6

```
void Config_INT6_Ext(void)
{
  P3IF &= ~(1<<6); // RAZ INT6 Pending Flag (P3IF page 177)
  P3IF |= (1<<2);  // Déclenchement INT6 sur front montant
  EIE2 |= (1<<4);  //EX6 = 1 Autorisation de l'interruption INT6
}
```

Figure 12.12. EIE2: Extended Interrupt Enable 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EXVLD	ES1	EX7	EX6	EADC1	ET4	EADC0	ET3	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE7

Bit4: EX6: Enable External Interrupt 6.
This bit sets the masking of External Interrupt 6.
0: Disable External Interrupt 6.
1: Enable interrupt requests generated by the External Interrupt 6 input pin.

Rappel: Procédure de mise en place d'une interruption – Etape 4

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**
3. Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité.
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**

4

6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

La solution Interruption avec INT6 – Etape 4

Etape 4 : Gestion de la priorité de l'interruption (par rapport aux autres programmes d'interruptions)

```
void Config_INT6_Ext(void)
{
    P3IF &= ~(1<<6); // RAZ INT6 Pending Flag (P3IF page 177)
   P3IF |= (1<<2);  // Déclenchement INT6 sur front montant

   EIE2 |= (1<<4);   //EX6 = 1 Autorisation de l'interruption INT6
   EIP2 |= (1<<4);   // Facultatif - Priorité haute
        pour l'interruption INT6
   }
```

Figure 12.14. EIP2: Extended Interrupt Priority 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
PXVLD	EP1	PX7	PX6	PADC1	PT4	PADC0	PT3	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:

Bit4: PX6: External Interrupt 6 Priority Control.
This bit sets the priority of the External Interrupt 6.
0: External Interrupt 6 set to low priority level.
1: External Interrupt 6 set to high priority level.

Rappel: Procédure de mise en place d'une interruption en assembleur

1. Ecrire la routine d'interruption ISR (*Interrupt Service Routine*), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».
2. Placer dans la table de vecteurs d'interruption, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.**
3. Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité.
4. Configurer le bit (*Enable Flag*) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.**
5. Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (*Priority Control*) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

5

La solution Interruption avec INT6 – Etape 5

Etape 4 :Validation Globale

```
48 //-------------------------------------
49 void main (void) {
50
51 //**************************************
52 // CONFIGURATIONS GLOBALES
53       Init_Device();
54 //**************************************
55 // CONFIGURATION des périphériques
56
57       Config_INT_Ext();
58 //**************************************
59 //  Séquence Démarrage
60       Phase_Demarrage();
61       EA = 1;
62 //**************************************
63 //Séquence usage
64       while(1)
65       {
66 //          Phase_Usage();
```

La validation globale des interruptions se fait de préférence au niveau «Main»

La solution Interruption avec INT6 - Main

```
while(1)
{
  EA = 0; // Section de code critique
  compteur_logiciel++;
  if (compteur_logiciel >= 1024) compteur_logiciel = 0;
  P7 = compteur_logiciel >>2;
  EA = 1;  // Fin Section de code critique
  Delay(1);

}
```

La variable compteur_logiciel est aussi manipulée dans le programme d'interruption.
Aussi, pour éviter tout risque « d'interférence » on interdit l'exécution des interruptions durant la manipulation de compteur_logiciel dans le « main »
On appelle cette portion de code, une section critique
Cette dévalidation des interruptions ne suspend pas leur détection, celles-ci seront exécutées dès la revalidation des interruptions.

La solution Interruption avec INT6 – code ISR

Le code de l'interruption INT6

```
//*******************************************************************
void ISR_INT6() interrupt 18
{
    VISU_Temoin_INT6 = 1;  // Facultatif - Signal Témoin INT6  pour vérification
                           // sur oscilloscope
    P3IF &=~0x40;          // Clear FLAG IE6


  compteur_logiciel = 0;
  P7 = compteur_logiciel >>2; //

    VISU_Temoin_INT6 = 0;
}
```

Remise à zéro du drapeau d'interruption INT6 (pending flag)
Son omission entraine la répétition à l'infini de l'interruption INT6!

LIVE AND DISCOVER

Contact

François JOLY
Tél. : 04 72 43 13 36
francois.joly@cpe.fr
www.cpe.fr

- CPE Lyon – Sciences du Numérique
- Domaine Scientifique de la Doua
- 43, bd du 11 novembre 1918 – Bâtiment Hubert Curien
- B.P. 2077 – 69616 Villeurbanne cedex – France