

# Electronique numérique programmée

Mise en œuvre et programmation des micro-contrôleurs

## **Objectifs**

*Mettre en œuvre un circuit d'entrée / sortie d'un système micro-contrôleur*

*Réaliser un organigramme de programmation d'un système micro-contrôleur*

*Rédiger en assembleur un programme à partir d'un organigramme*

## **Pré-requis**

- *Electronique numérique, série 8*
- *Principe organigramme*

## **Savoirs associés**

- *Structure d'un système programmable*
- *Organisation de la zone adressable*
- *Mécanisme des interruptions*
- *Programmation en assembleur*

# Sommaire

## **I. Introduction**

## **II. Structure d'un système programmable**

- 1. Architecture des systèmes à base de micro-processeur**
- 2. Architecture des systèmes à base de micro-contrôleur**

## **III. Choix d'un micro-contrôleur**

## **IV. Développement de système à base de micro-contrôleur**

- 1. Organisation du développement (programmation seule)**
- 2. Organisation du développement (matériel et programmation)**

## **V. Etude d'un micro-contrôleur 16 bits : le 8XC196KC**

- 1. Architecture du 8XC196KC**
- 2. Zone adressable du micro-contrôleur**
- 3. Fichier de registre et Special Fonction Register du micro-contrôleur**
- 4. Ports d'entrées / sorties Tout Ou Rien**
- 5. Ports de sorties à Modulation de Largeur d'Impulsions (PWM)**
- 6. Interruptions du micro-contrôleur**
- 7. Ports d'entrées de conversion analogique / numérique**

## **Travail personnel**

### **Autocorrection**

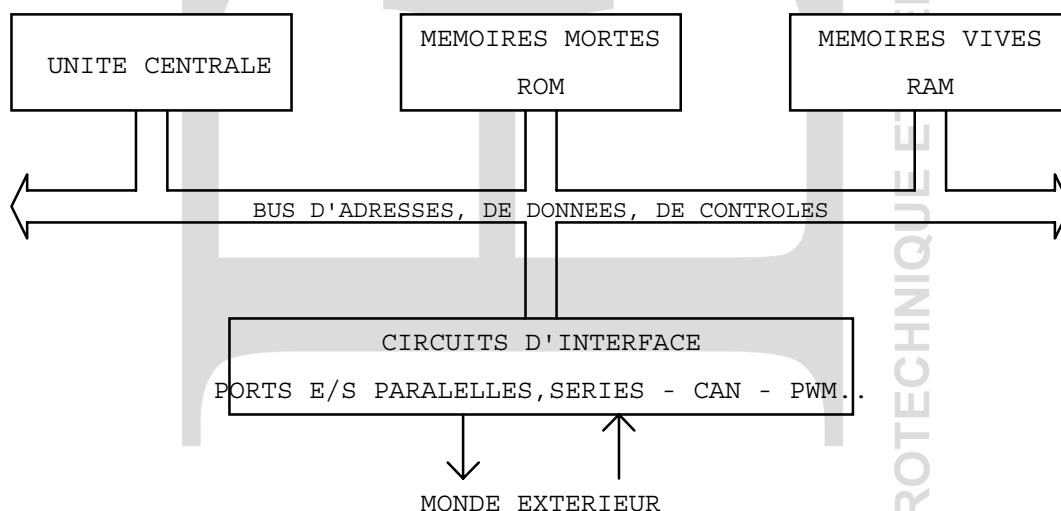
## I. Introduction

Le micro-contrôleur est né lorsque les technologies d'intégration ont eu suffisamment progressé pour permettre sa fabrication et aussi parce que très souvent, dans des applications domestiques ou industrielles, on a besoin d'un système "intelligent" ,complet, puissant, facile à mettre en oeuvre et d'une grande souplesse d'emploi ; le tout dans un seul boîtier. Il permet dans un seul boîtier d'avoir tous les éléments nécessaire à la mise en oeuvre d'un automatisme industriel. Il ne reste qu'à l'interfacer avec les différents capteurs et préactionneurs du process à commander. Grâce à l'arrivée des micro-contrôleurs, des cartes qui contenaient des dizaines de circuits intégrés logiques se sont vues réduites à un seul boîtier.

## II. Structure d'un système programmable

### 1. Architecture des systèmes à base de micro-processeur

Le micro-processeur a besoin pour fonctionner d'un environnement minimum : celui-ci est au moins composé du processeur lui-même (exécutant les tâches à accomplir) des mémoires ROM (contenant le programme à exécuter) et RAM (stockant des données temporairement) ainsi que des interfaces avec l'extérieur (clavier, lecteur de carte, écran, contrôleur d'IGBT etc...). L'ensemble des boîtiers dialogue par des bus (ensemble de fils) qu'il faut implanter sur des circuits imprimés, ce qui représente donc un volume important et un coût élevé.



### 2. Architecture des systèmes à base de micro-contrôleur

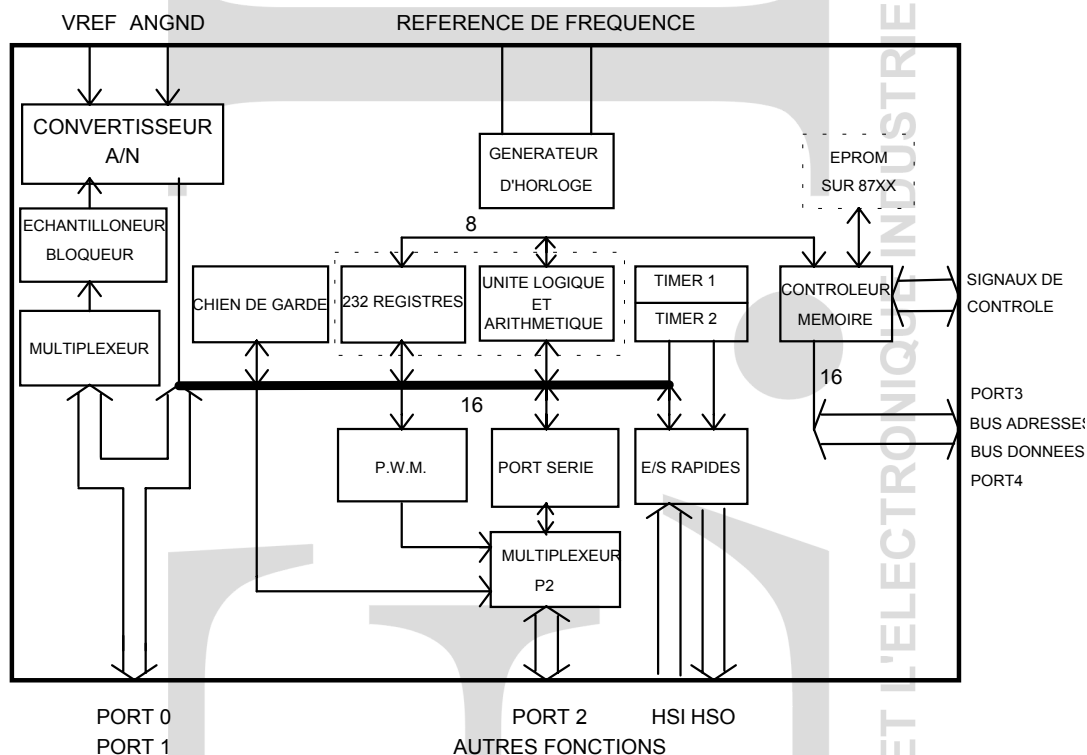
Les fabricants de micro-contrôleurs intègrent autant que possible dans un seul boîtier l'ensemble des fonctions de la structure présentée ci-dessus. Cette intégration poussée est généralement réalisée en technologie HCMOS qui permet une diminution de la consommation énergétique, une simplification du tracé du circuit imprimé ( nombre réduit de boîtiers ), une augmentation de la fiabilité du système ainsi qu'une réduction globale du coût.

Les micro-contrôleurs sont utilisés dans de nombreux équipements que nous cotoyons tous les jours (four micro-onde, téléphone portable, perceuse électronique, imprimante etc...) car leurs coûts sont extrêmement faibles.

Voici un exemple de structure de micro-contrôleur : la famille 196.

L'unité centrale est organisée autour du noyau dit 196 qui est le même pour tous les micro-contrôleurs de la famille MCS96. Elle dispose d'un bus interne 16 bits sur lequel est connecté le bloc de 232 registres (cases mémoire RAM) ainsi qu'un " Watch Dog " ( chien de garde )

destiné, sur anomalie de fonctionnement, à relancer le programme automatiquement à son point d'entrée.



### III. Choix d'un micro-contrôleur

Il existe plusieurs micro-contrôleurs fabriqués par Intel, Motorola, Hitachi, Nec, Texas Instruments etc...

Le choix d'un modèle dépend essentiellement de l'application :

- Nombre d'entrées/sorties Tout Ou Rien.
- Liaison d'entrée/sortie série.
- Conversion analogique numérique et numérique analogique.
- Entrées/sorties rapides sortie spéciales (M.L.I., horodatées etc...).
- Mémoire RAM, ROM, EPROM interne ou externe, sa taille.
- Vitesse de l'horloge. Temps d'exécution d'une multiplication, d'une division.
- Bus de données 8 bits/16 bits.
- Type de boîtier.

Puis, on se posera des questions au sujet de l'aide au développement:

- Les logiciels de programmation (Assembleur, C etc...).
- Les émulateurs pour la mise au point des applications.
- Les évolutions prévisibles du composant, son prix, les sources.

Pour répondre à ces questions les constructeurs ont développé non pas des micro-contrôleurs isolés mais des familles de micro-contrôleurs.

Les micro-contrôleurs de la même famille ont :

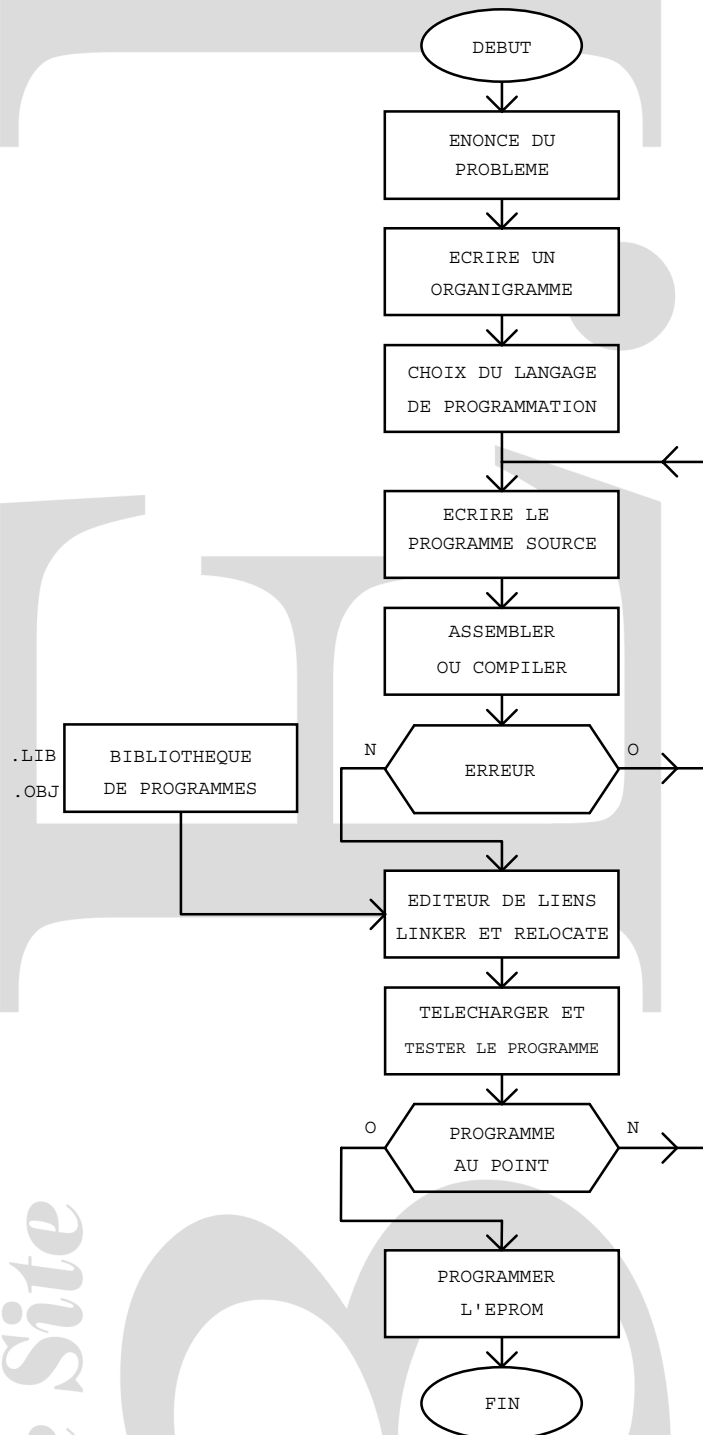
- Une unité centrale commune.
- Une architecture commune plus ou moins variée.
- Les mêmes outils de programmation et de développement.
- Différents types de circuits périphériques intégrés.

Exemple famille MCS 96 de INTEL

DEVICES	ROM/ EPROM	RAM	TIMER COUNTER	A/D CHANELS	I/O PINS	I/O TYPE	SERIAL PORTS	SPEED ( MHz )	PROCESS
<b>8098 PRODUCE LINE</b>									
8098	ROMLESS	232	2	4	32	HSI/O	1	12	HMOS
8398	8K ROM	232	2	4	32	HSI/O	1	12	HMOS
8798	8K EPROM	232	2	4	32	HSI/O	1	12	HMOS
<b>80C198 PRODUCE LINE</b>									
80C198	ROMLESS	232	2	4	34	HSI/O	1	12	CHMOS
83C198	8K ROM	232	2	4	34	HSI/O	1	12	CHMOS
87C198	8K OPT	232	2	4	34	HSI/O	1	12	CHMOS
<b>8096BH PRODUCE LINE</b>									
8096BH	ROMLESS	232	2	NO	48	HSI/O	1	12	HMOS
8396BH	8K ROM	232	2	NO	48	HSI/O	1	12	HMOS
8797BH	8K EPROM	232	2	8	48	HSI/O	1	12	HMOS
<b>8097JF PRODUCE LINE</b>									
8097JF	ROMLESS	232	2	8	48	HSI/O	1	12	HMOS
8397JF	16K ROM	232	2	8	48	HSI/O	1	12	HMOS
8797JF	16K OPT	232	2	8	48	HSI/O	1	12	HMOS
<b>80C196 PRODUCE LINE</b>									
80C196KB	ROMLESS	232	2	8	48	HSI/O	1	10, 12	CHMOS
83C196KB	8K ROM	232	2	8	48	HSI/O	1	10, 12	CHMOS
87C196KB	8K EPROM	232	2	8	48	HSI/O	1	10, 12	CHMOS
80C196KC	ROMLESS	488	2	8	48	HSI/O	1	16	CHMOS
83C196KC	16K ROM	488	2	8	48	HSI/O	1	16	CHMOS
87C196KC	16K EPROM	488	2	8	48	HSI/O	1	16	CHMOS
<b>80C196KR PRODUCE LINE</b>									
80C196KR	ROMLESS	488	2	8	56	EPA	2	16	CHMOS
83C196KR	16K ROM	488	2	8	56	EPA	2	16	CHMOS
87C196KR	16K EPROM	488	2	8	56	EPA	2	16	CHMOS

## IV. Développement de système à base de micro-contrôleur

### 1. Organisation du développement (programmation seule)



On ne peut pas parler de micro-contrôleur sans aborder les logiciels de programmation et les matériels permettant de développer le composant. Pour réaliser les programmes exécutables, on

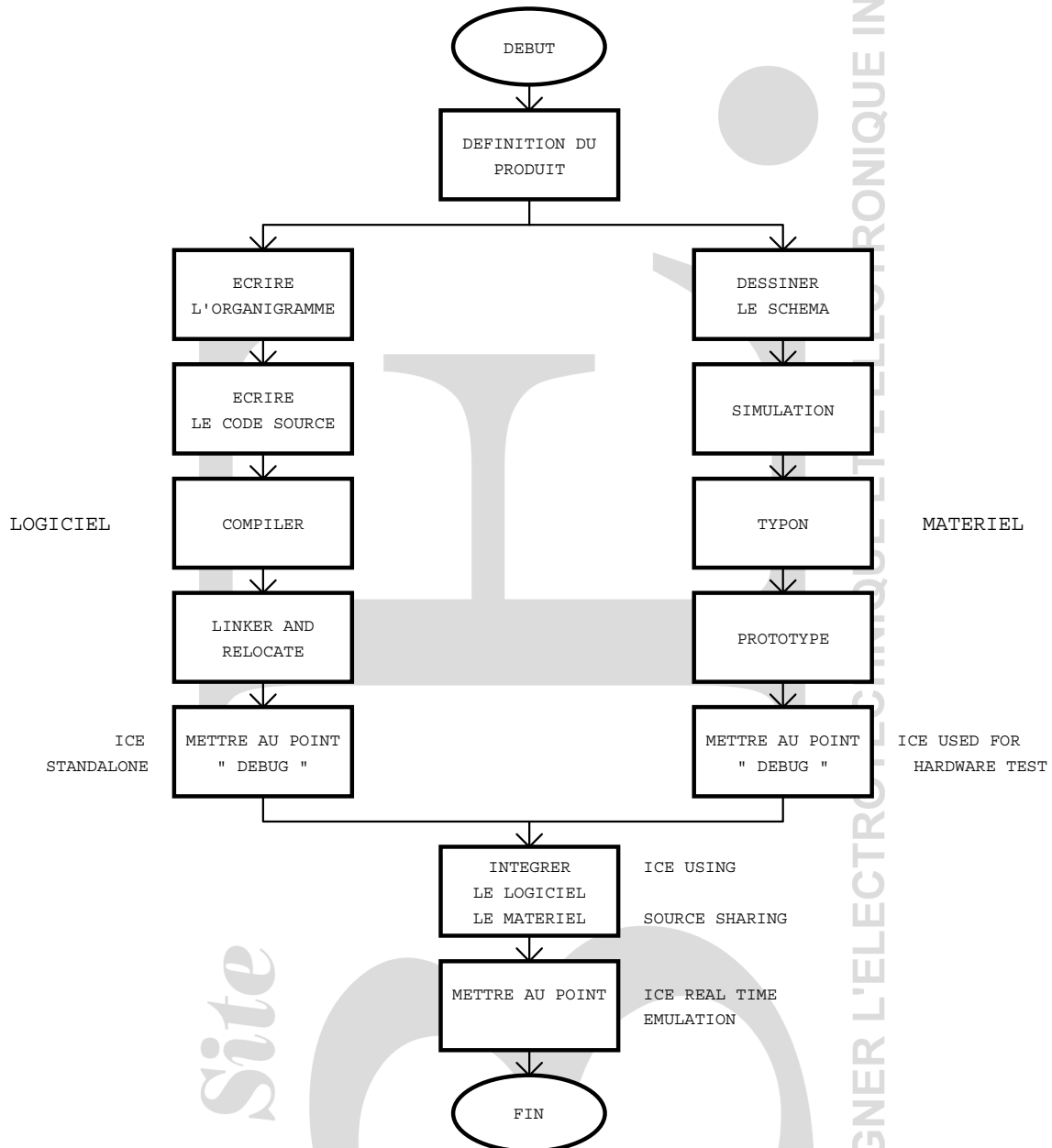
utilisera l'assembleur ou le compilateur C (édités par le constructeur ou une autre société). Pour étudier le micro-contrôleur, il existe une carte d'évaluation développée par le constructeur.

## 2. Organisation du développement (matériel et programmation)

Pour cette solution, on développe conjointement la carte micro-contrôleur et le logiciel.

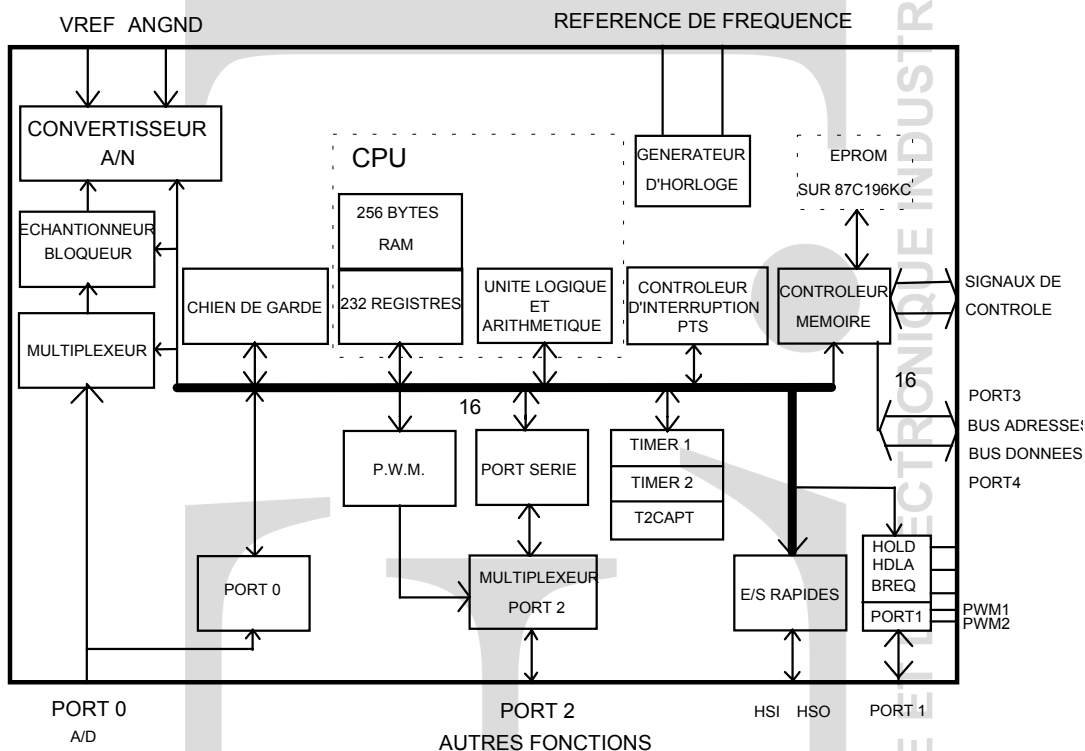
Le développement de la carte et du logiciel associé comprend trois phases :

1. Développement de la carte et du programme simultanément.
2. Association des deux parties matérielle et logicielle.
3. Utilisation d'un émulateur pour tester cet ensemble.



## V. Etude d'un micro-contrôleur 16 bits : le 8XC196KC

### 1. Architecture du 8XC196KC



A partir de cette structure, on constate que le 8XC196KC dispose d'un certain nombre d'entrées et de sorties:

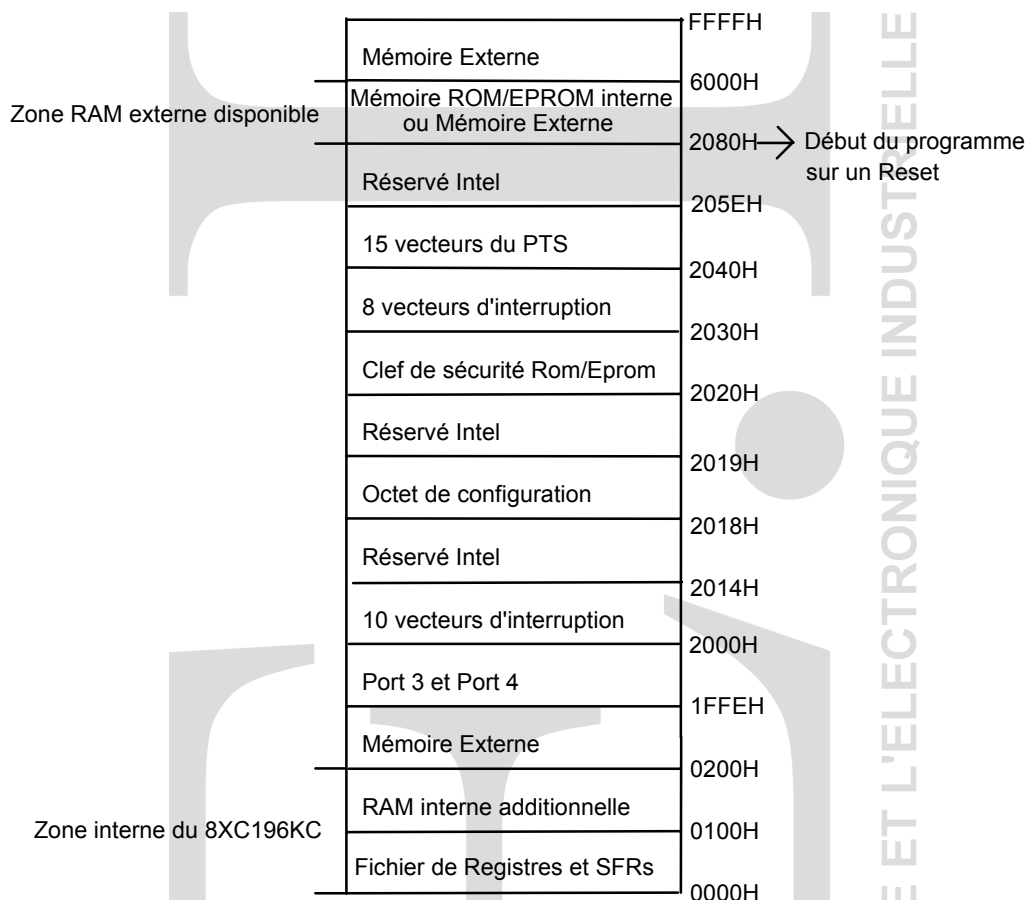
- 5 ports d'entrées/sorties parallèles sur 8 bits (Port 0, Port 1, Port 2, Port 3 et Port 4).
- 2 à 4 entrées rapides HSI.
- 6 à 4 sorties rapides HSO.
- 8 entrées Convertisseur Analogique/Numérique multiplexées sur 8 ou 10 bits.
- 3 sorties PWM (PWM0, PWM1, PWM2).
- 1 port de communication série.
- 26 registres spéciaux ( SFRs ).
- 232 registres (cases mémoire).
- 256 octets de mémoire type RAM reconfigurable en registre.
- 2 compteurs ( Timer 1 et Timer 2 ) utilisables avec les Entrées/Sorties rapides.
- 4 Timers logiciels utilisés pour générer les événements.
- 1 chien de garde utilisé pour verrouiller l'unité centrale dans certain cas de figure.
- 1 contrôleur de mémoire externe.
- 28 sources d'interruption.

### 2. Zone adressable du micro-contrôleur

Le micro-contrôleur 8XC196KC possède 16 bits d'adresse. Il peut donc adresser 64 Koctets de mémoire. Voici la répartition des 64 Koctets de mémoires adressables par le micro-contrôleur 8XC196KC sur la carte INTEL EVB196 :

- La mémoire interne du 8XC196KC apparaît en bas de  $0000_H$  à  $01FF_H$  (2 x 256 cases).
- La mémoire externe apparaît de  $2000_H$  à  $5FFF_H$  (4096 cases dont certaines affectées).
- Les ports 3 et 4 sont situés au adresses  $1FFE_H$  et  $1FFF_H$ .





### 3. Fichier de registre et Special Fonction Register du micro-contrôleur

#### 3.1. Fichier de registre

L'ensemble de la mémoire RAM compris entre 0018<sub>H</sub> à 00FF<sub>H</sub> constitue le fichier de registre (Register File). Ils sont 232 et peuvent être accédés en 8 bits (BYTE) ou 16 bits (WORD) ou 32 bits (DOUBLE WORD). Ces registres (cases mémoires) sont essentiellement utilisés par le processeur comme emplacements disponibles pour un stockage temporaire.

Les adresses 0018<sub>H</sub> et 0019<sub>H</sub> sont nécessairement réservées au pointeur de pile s'il est utilisé (nécessaire pour les interruptions). Elles peuvent être utilisées en tant que mémoire si le pointeur de pile n'est pas utilisé.

En plus de ces 232 octets de registre, le 8XC196KC possède 256 octets de mémoires RAM supplémentaires (0100<sub>H</sub> à 01FF<sub>H</sub>). Ces mémoires peuvent être utilisés par le processeur en tant que registre.

#### 3.2. Registres à fonctions spéciales SFRs

De l'adresse 0000<sub>H</sub> à 0017<sub>H</sub>, nous trouvons des registres contrôlant la gestion des entrées/sorties du micro-contrôleur. Ces registres s'appellent SFRs.

Toutes les opérations d'entrées et de sorties du 8XC196KC sont contrôlées par ces SFRs (exceptés les PORT3 et PORT4 qui se trouvent aux adresses 1FFE<sub>H</sub> et 1FFF<sub>H</sub>).

Les SFRs sont situés dans trois zones de mémoire définies par trois fenêtres horizontales (HWINDOW = horizontale window).

- HWINDOW 0 (WSR = 0) pour la première colonne des SFRs.

- HWINDOW 1 (WSR = 1) pour la deuxième colonne des SFRs.
- HWINDOW 15 (WSR = 15) pour la troisième colonne des SFRs.

TABLEAU DES SFRs

19H	SP(HI)	SP(HI)	SP(HI)	SP(HI)	SP(HI)
18H	SP(LO)	SP(LO)	SP(LO)	SP(LO)	SP(LO)
17H	IOS2	PWM0_CONTROL	PWM1_CONTROL	PWM0_CONTROL	IOS2
16H	IOS1	IOC1	PWM2_CONTROL	IOC1	IOS1
15H	IOS0	IOC0	RESERVED	IOC0	IOS0
14H	WSR	WSR	WSR	WSR	WSR
13H	IMASK1	IMASK1	IMASK1	IMASK1	IMASK1
12H	IPEND1	IPEND1	IPEND1	IPEND1	IPEND1
11H	SP_STAT	SP_CON	RESERVED	SP_CON	SP_STAT
10H	PORT2	PORT2	RESERVED	RESERVED	RESERVED
0FH	PORT1	PORT1	RESERVED	RESERVED	RESERVED
0EH	PORT0	BAUD_RATE	RESERVED	RESERVED	RESERVED
0DH	TIMER2(HI)	TIMER2(HI)	RESERVED	T2CAPTURE(HI)	T2CAPTURE(HI)
0CH	TIMER2(LO)	TIMER2(LO)	IOC3	T2CAPTURE(LO)	T2CAPTURE(LO)
0BH	TIMER1(HI)	IOC2	RESERVED	IOC2	TIMER1(HI)
0AH	TIMER1(LO)	WATCHDOG	RESERVED	WATCHDOG	TIMER1(LO)
09H	INT_PEND	INT_PEND	INT_PEND	INT_PEND	INT_PEND
08H	INT_MASK	INT_MASK	INT_MASK	INT_MASK	INT_MASK
07H	SBUF(RX)	SBUF(TX)	PTSSRV(HI)	SBUF(TX)	SBUF(RX)
06H	HSI_STATUS	HSO_COMMAND	PTSSRV(LO)	HSO_COMMAND	HSI_STATUS
05H	HSI_TIME(HI)	HSO_TIME(HI)	PTSSEL(HI)	HSO_TIME(HI)	HSI_TIME(HI)
04H	HSI_TIME(LO)	HSO_TIME(LO)	PTSSEL(LO)	HSO_TIME(LO)	HSI_TIME(LO)
03H	AD_RESULT(HI)	HSI_MODE	AD_TIME	HSI_MODE	AD_RESULT(HI)
02H	AD_RESULT(LO)	AD_COMMAND	RESERVED	AD_COMMAND	AD_RESULT(LO)
01H	ZERO_REG(HI)	ZERO_REG(HI)	ZERO_REG(HI)	ZERO_REG(HI)	ZERO_REG(HI)
00H	ZERO_REG(LO)	ZERO_REG(LO)	ZERO_REG(LO)	ZERO_REG(LO)	ZERO_REG(LO)
	<b>HWINDOW 0</b>	<b>HWINDOW 0</b>	<b>HWINDOW 1</b>	<b>HWINDOW 15</b>	<b>HWINDOW 15</b>
	<b>Lecture</b>	<b>Ecriture</b>	<b>Lecture/Ecrit</b>	<b>Lecture</b>	<b>Ecriture</b>

Chaque fenêtre est accessible selon le contenu du registre WSR (14<sub>H</sub>).

**EXEMPLE** : Si on veut accéder au registre PWM1\_CONTROL situé dans la fenêtre 1 (HWINDOW 1 en écriture), on doit d'abord charger le contenu de WSR à 01<sub>H</sub> puis charger le registre PWM1\_CONTROL à la valeur voulue.

```
LDB WSR,#01H ;pour accéder à la fenêtre 1
LDB PWM1_CONTROL,#VALEUR OPERANDE ;pour charger PWM1_CONTROL
```

### REMARQUES

- Certains registres ne sont accessibles que par mot (word), par exemple HSI\_TIME (04<sub>H</sub>).
- Ne pas utiliser les registres réservés.
- Dans les fenêtres 0 et 15, suivant que l'on est en écriture ou en lecture nous ne trouvons pas nécessairement les mêmes registres.

### QUELQUES COMPLEMENTS D'INFORMATION

- Le mot ZERO\_REG ( 00H ) est chargé avec 0000H de façon permanente.

- Les octets IOC0, IOC1, IOC2, IOC3 permettent de configurer certains périphériques du  $\mu\text{C}$ .
- Les octets IOS0, IOS1, IOS2 donnent l'état de ces mêmes périphériques (ex : conversion CAN terminée).
- Pour chaque périphérique, nous reviendrons sur l'état de ces registres SFRs.

**Vérifier l'état des registres SFRs** concernés, avant d'utiliser les E/S.

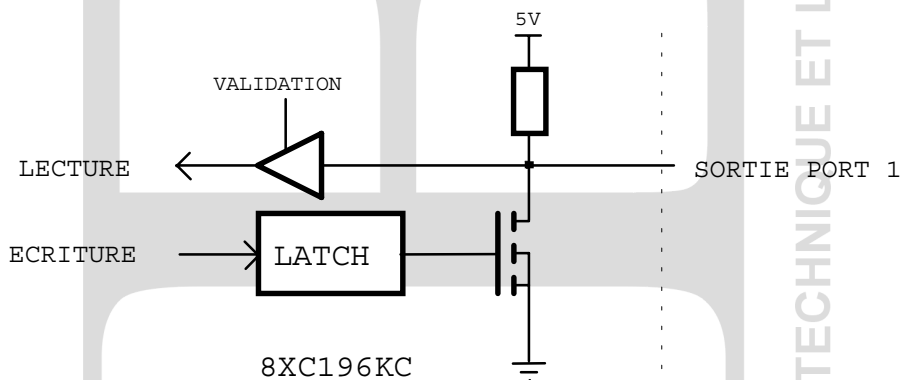
#### 4. Ports d'entrées / sorties Tout Ou Rien

Nous avons 5 ports d'entrées/sorties parallèles à 8 bits (ports 0, 1, 2, 3 et 4). Il existe des ports qui sont à entrée seule, à sortie seule et des ports qui sont bidirectionnels. Les broches où se trouvent les ports d'entrées/sorties parallèles sont souvent utilisées avec d'autres fonctions (le port 0 est sur les mêmes broches que les entrées analogiques par exemple).

Le **PORT 0** est un port d'entrée (à lecture seule). Son contenu est accessible à l'adresse  $0E_H$  des SFRs de la fenêtre HWINDOW 0. En plus de cette fonction entrée logique, ce port peut être utilisé en entrée analogique. C'est donc un port multifonctions. Les niveaux logiques sont :

- Ventrée  $< 0.8V$  Niveau logique 0
- Ventrée  $> 2.2V$  Niveau logique 1

Le **PORT 1** fonctionne quand à lui totalement en mode bidirectionnel (soit en sortie, soit en entrée). Son contenu est accessible à l'adresse  $0F_H$  des SFRs de la fenêtre HWINDOW 0 (en écriture pour la sortie, en lecture pour l'entrée). Les niveaux d'entrée sont les mêmes que pour le port 0. Voici le schéma de ce port pour l'entrée et la sortie.



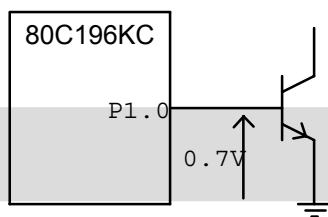
**ATTENTION :** Pour les ports bidirectionnels, il faut adopter quelques règles:

**En entrée** si l'ensemble des entrées est relié à la masse, le courant de 7mA par sortie risque d'être trop important pour le boîtier (  $7\text{mA} \times 8 = 56\text{mA}$  ).

Solution matérielle : mettre une résistance en série de  $1K\Omega$ .

Solution logicielle : écrire 1 dans le port 1, lorsqu'il est utilisé en entrée.

**En sortie** dans certains cas de figure, on peut avoir un niveau électrique sur la broche de sortie différent du contenu du registre du port 1 en lecture. Par exemple avec le montage ci-dessous si on écrit dans le port 1 à l'adresse  $0F_H$  des SFRs le mot  $00000001_B$ . le transistor est bien saturé, mais si on lit le contenu du port 1, on aura  $00000000_B$  à cause du niveau électrique d'entrée du transistor bipolaire.

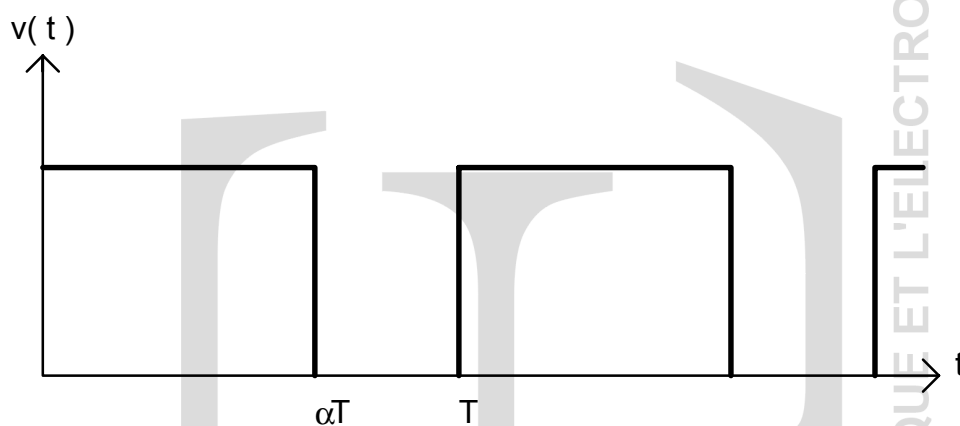


Le **PORT 2** est un port multifonctions que nous n'étudierons pas.

Les **PORTS 3 et 4** sont bidirectionnels et sont les seuls à ne pas être contrôlés par les SFRs. Ils sont accessibles aux adresses  $1FFE_H$  et  $1FFF_H$ . L'utilisation de ces 2 ports est particulière puisqu'ils sont multiplexés avec les bus d'adresse et de données. Nous n'étudierons pas ces deux ports

### 5. Ports de sorties à Modulation de Largeur d'Impulsions (PWM)

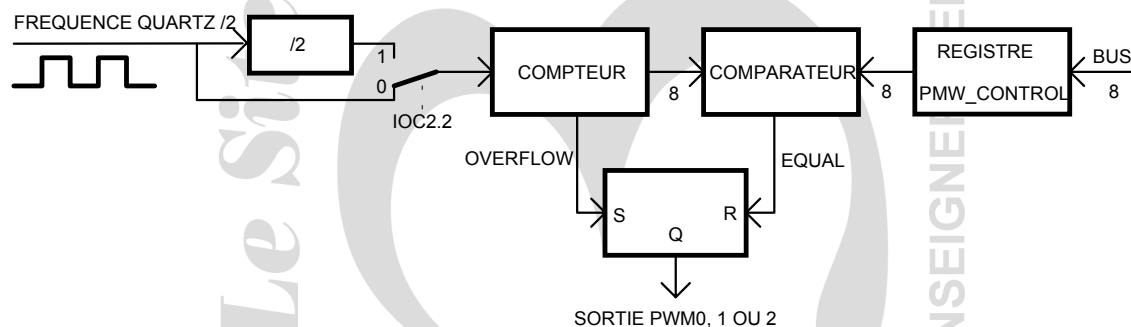
On souhaite réaliser un signal rectangulaire de fréquence fixe et de rapport cyclique variable.



Trois solutions sont envisageables :

- En utilisant un port de sortie parallèle dont on change l'état en agissant sur deux temporisateurs complémentaires ( $T_1$  pour  $\alpha T$ ;  $T_2$  pour  $(T - \alpha T)$ ). Avec un tel système, nous pouvons utiliser ce signal dans une large gamme de fréquence et de rapport cyclique, mais ce choix est lourd car il faut deux temporisateurs, du calcul mobilisant le processeur pour exécuter le programme.
- En utilisant une sortie rapide HSO du micro-contrôleur. Nous ne l'étudierons pas.
- En utilisant une des trois sorties PWM du micro-contrôleur.

Etudions cette dernière solution. Les sorties PWM sont au nombre de trois: PWM0, PWM1, PWM2. Le synoptique fonctionnel est donné ci-dessous.

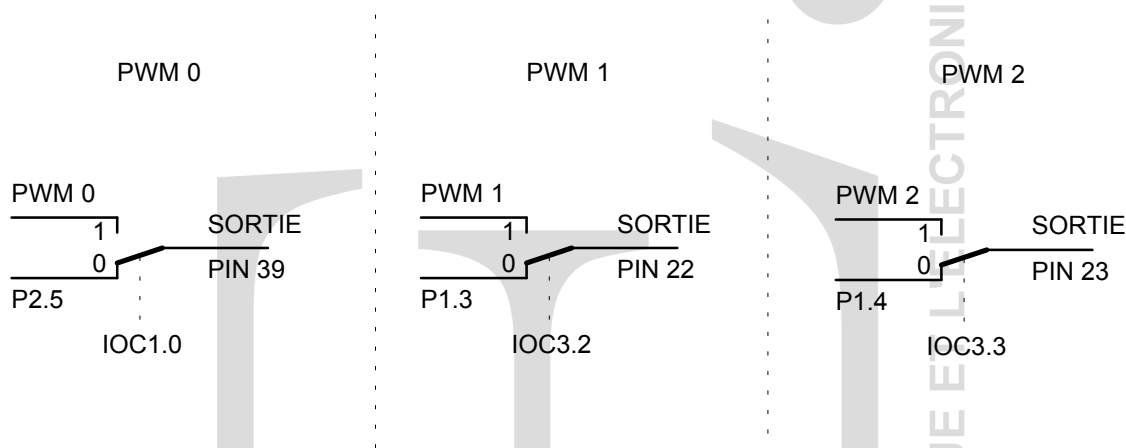


La période  $T$  est imposée par le quartz. Si la fréquence du quartz est de 16 MHz :

- Quand IOC2.2 = 0, la période PWM est  $T = \frac{256}{8 \times 10^6} = 32 \mu\text{s}$  ou 31,25 KHz
- Quand IOC2.2 = 1, la période PWM est  $T = \frac{2 \times 256}{8 \times 10^6} = 64 \mu\text{s}$  ou 15,625 KHz

**FONCTIONNEMENT :** Le compteur 8 bits s'incrémente à chaque impulsion d'horloge. Lorsque le compteur est égal à 00H ( overflow ), la sortie PWM passe à 1. Quand le compteur atteint la valeur chargée dans le registre PWM\_CONTROL, la sortie du PWM passe à 0. Lorsque le compteur est à nouveau en dépassement, la sortie PWM bascule de nouveau à 1. Le signal de sortie de PWM est de fréquence fixe et rapport cyclique variable avec le contenu du registre PWM\_CONTROL. Le rapport cyclique vaut PWM\_CONTROL/256.

Pour accéder aux sorties PWM une sélection est nécessaire par un bit de direction :

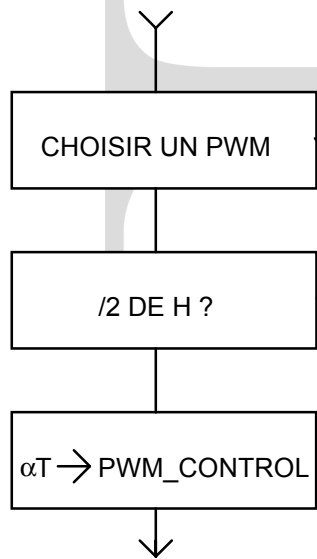


A l'aide du registre de contrôle concerné, on dirige la sortie PWM correspondante vers la sortie.

**Remarques:** Les 3 PWM utilisent le même compteur de 8 bits. Donc, ces trois sorties passent en même temps au niveau 1 logique. Le bit de division de l'horloge IOC2.2 est donc actif pour les trois PWM. La nouvelle valeur de PWM\_CONTROL ne sera prise en compte qu'après le passage en overflow du compteur.

Si le contenu de PWM\_CONTROL = 0FF<sub>H</sub>, alors la sortie du PWM reste à l'état haut de 00<sub>H</sub> à 0FE<sub>H</sub> et passe à zéro de 0FF<sub>H</sub> à 00<sub>H</sub>. Le rapport cyclique maximal est donc de 99,6 %

INITIALISATION D'UNE PWM:



Si on choisit le PWM 1, voici le programme:

```
CLRB WSR ;pour accéder à HWINDOW 0  
(mettre à 0 WSR pour sélectionner la fenêtre HWINDOW 0)
```

```
LDB IOC2,#00000100B;/2 de Horloge  
(mettre à 1 le bit 2 du registre IOC2 pour sélectionner la  
division par deux de l'horloge)
```

```
LDB WSR,#01;pour accéder à HWINDOW 1  
(mettre à 1 WSR pour sélectionner la fenêtre HWINDOW 1)
```

```
LDB IOC3,#00000100B;choix du PWM1  
(mettre à 1 le bit 2 du registre IOC3 pour envoyer la sortie  
PWM1 vers la patte 22 du micro-contrôleur)
```

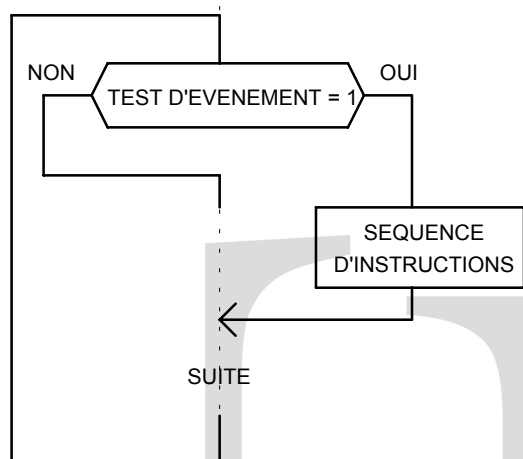
```
LDB PWM1_CONTROL,#80H ;α=0,5  
(charger PWM1_CONTROL par la valeur hexadécimale  
80 pour obtenir un rapport cyclique de 50 %)
```

## 6. Interruptions du micro-contrôleur

En automatique la prise en compte d'événements (changement d'état d'une entrée) peut être envisagée selon deux approches :

**METHODE SCRUTATIVE** : Dans la suite d'instructions du programme on insère un test puis en fonction du résultat on réalise la séquence d'instructions correspondante.

Le micro-contrôleur scrute périodiquement l'état du test lorsqu'il rencontre l'instruction dans son programme.

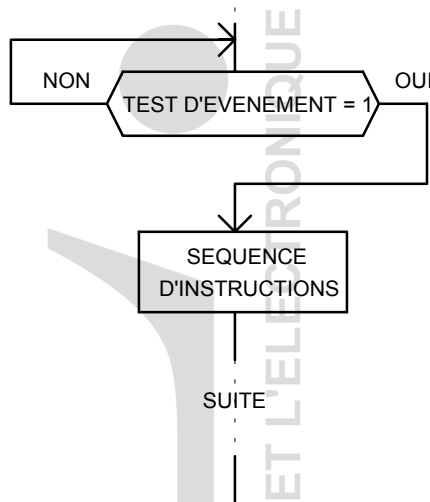


La séquence d'instructions sera effectuée lorsque le micro-contrôleur détectera un test vrai.

Problèmes liés à la méthode

- ◆ Non synchronisation entre l'apparition de l'événement et son test avec risque de non prise en compte d'un événement fugitif.
- ◆ Retard variable entre l'apparition de l'événement et la réalisation de la séquence d'instructions correspondante.

Le micro-contrôleur scrute en permanence l'état du test.



La séquence d'instruction sera effectuée lorsque le micro-contrôleur détectera un test vrai.

Problèmes liés à la méthode

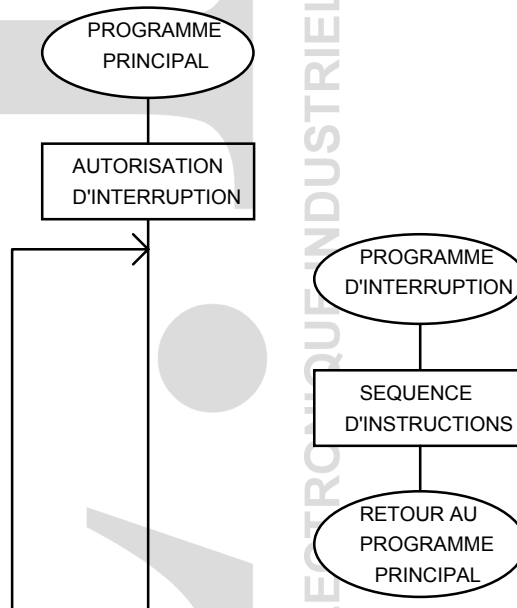
- ◆ Le micro-contrôleur ne fait rien durant l'attente de l'événement.

**Les limites de ces méthodes sont dues à leur caractère "soft". Une meilleure méthode passe par une solution "hard" du problème.**

**METHODE INTERRUPTIVE** : Le programme est séparé en deux segments autonomes. Seul le programme principal est exécuté tant que l'événement ne se produit pas. Si l'événement arrive le micro-contrôleur doit alors interrompre le programme principal là où il se trouve pour exécuter le programme d'interruption. Lorsque ce dernier se terminera, le micro-contrôleur reviendra au programme principal à l'endroit où il avait été interrompu.

Dispositions constructives particulières :

- ◆ Il faut autoriser le micro-contrôleur à prendre en compte les interruptions.
- Lorsqu'une interruption arrive
- ◆ Il faut sauvegarder le contexte dans lequel se trouvait le micro-contrôleur avant d'aller exécuter le programme d'interruption.
  - ◆ Il faut informer le micro-contrôleur de l'adresse du programme d'interruption.
  - ◆ Il faut exécuter le programme d'interruption.
  - ◆ Il faut restituer le contexte dans lequel se trouvait le micro-contrôleur avant d'aller exécuter le programme d'interruption.
  - ◆ Il faut récupérer l'adresse de reprise du programme principal qui avait été sauvegardée.
  - ◆ Il faut continuer à exécuter le programme principal.



Il faut pouvoir étendre ce principe à différents types d'événements et gérer les priorités entre les événements.

**INTERRUPTIONS DU 80C196KC : SOURCE ET VECTEUR D'INTERRUPTION**

Le 80C196KC dispose de 28 sources d'interruptions (événement susceptible d'interrompre l'exécution d'un programme) et seulement de 16 vecteurs (adresse de branchement du sous programme d'interruption) utilisables pour lesquelles des niveaux de priorité ont été définis par le constructeur (vecteurs classés par ordre de priorité décroissant).

<b>SOURCES</b>	<b>VECTORS</b>
NON_MASQUABLE INTERRUPT	NMI
TIMER 2 CAPTURE	TIMER 2 CAPTURE
4 TH FIFO ENTRY	HSI FIFO 4
UNIMPLEMENTED OPCODE	UNIMPLEMENTED OPCODE
TRAP INSTRUCTION	SOFTWARE TRAP
EXTINT P2.2	EXTINT PIN
PORT 0.7	EXTINT
TI FLAG	TI FLAG
RI FLAG	SERIAL PORT
SOFTWARE TIMER 0	RI FLAG
SOFTWARE TIMER 1	SOFTWARE TIMER
SOFTWARE TIMER 2	
SOFTWARE TIMER 3	
RESET TIMER 2	
START A/D	
HSI_0 PIN	HSI_0 PIN
HSO LINE 0	HIGH SPED OUTPUT
HSO LINE 1	
HSO LINE 2	
HSO LINE 3	
HSO LINE 4	
HSO LINE 5	
HSI FIFO FULL	HSI FIFO FULL
HSI HOLDING REGISTER LOADED	HSI DATA AVAILABLE
A/D CONVERSION COMPLETE	A/D CONVERSION COMPLETE
TIMER 2 OVERFLOW ( 0000H or 8000H )	TIMER 2 OVERFLOW
TIMER 1 OVERFLOW	TIMER OVERFLOW



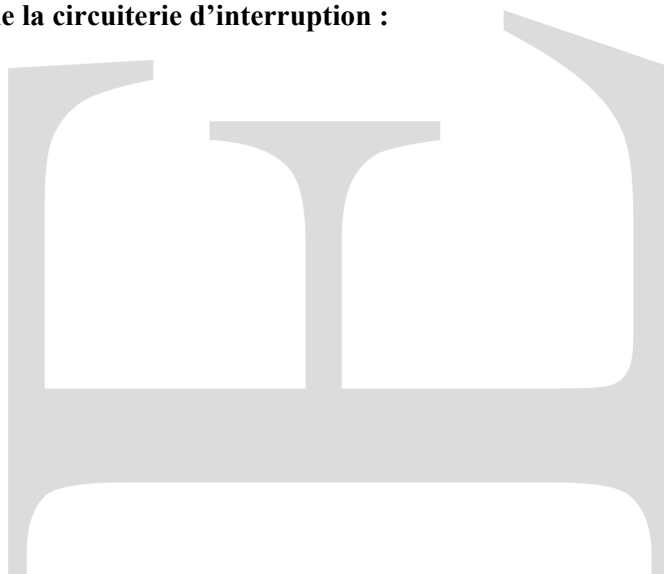
Ces interruptions sont mémorisées dans deux registres IPEND1 et INT\_PENDING de plus il est possible de les masquer (interdire leur prise en compte) grâce à deux registres IMASK1 et INT\_MASK. Il est également possible de masquer l'ensemble des interruption par I dans PSW.9.

	7	6	5	4	3	2	1	0
12H IPEND1:	NMI	HSI FIFO	EXT INT PIN	T2 OVFL	T2 CAPT	HSI FIFO 4 <sup>ème</sup>	RI	TI
13H IMASK1:								

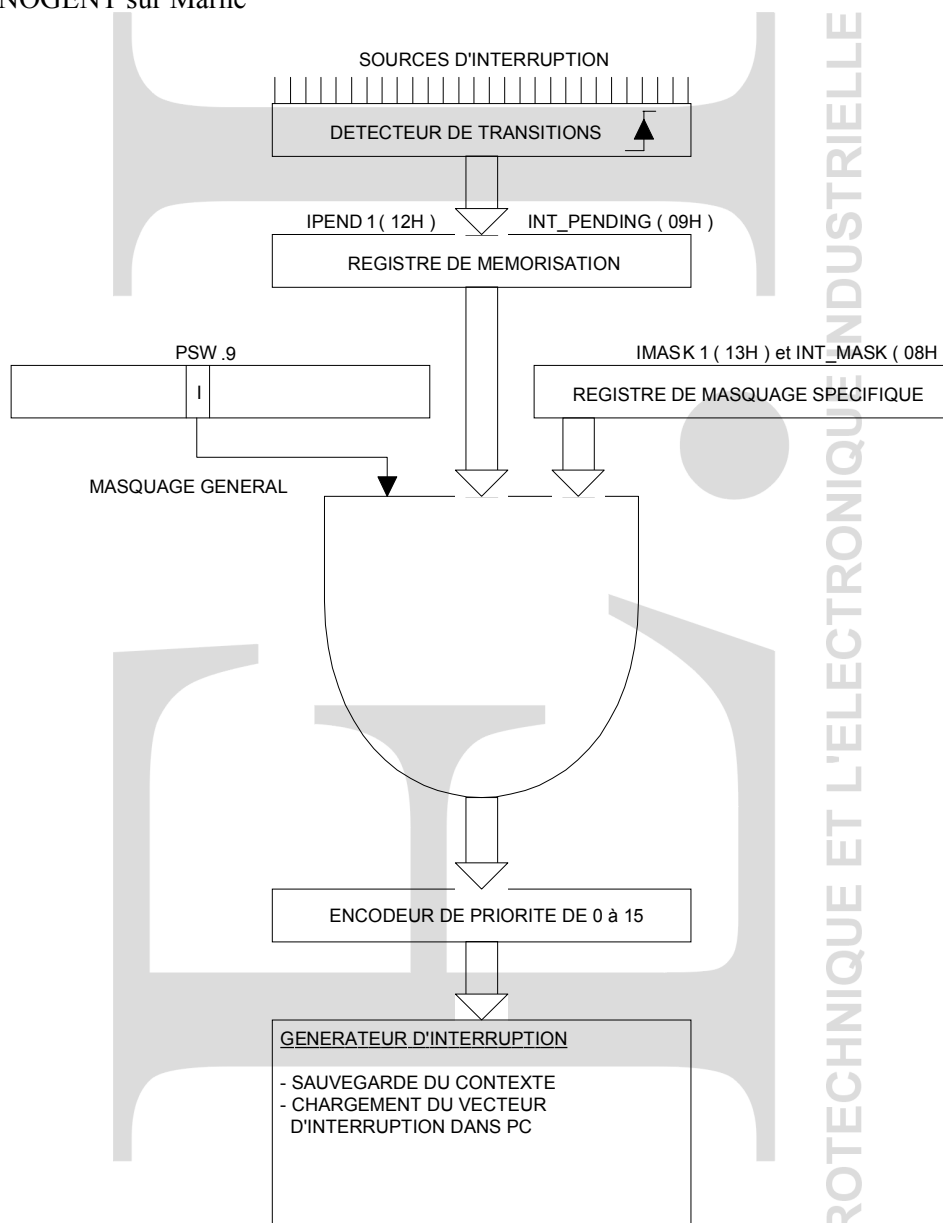
	7	6	5	4	3	2	1	0
09H INT_PENDING:	EXT	SERIAL	SOFT	HSI.0	HSO	HSI	A/D	TIMER
08H INT_MASK:	INT	PORT	TIMER	PIN	PIN	DATA	DONE	OVFL

Chaque vecteur utilise 2 adresses consécutives dans l'espace 2000<sub>H</sub>/200F<sub>H</sub> et 2030<sub>H</sub>/203F<sub>H</sub> dans lesquelles sera rangée par l'utilisateur l'adresse de sa routine d'interruption (ISR).

**Organisation de la circuiterie d'interruption :**



ENSEIGNER L'ELECTROTECHNIQUE ET L'ELECTRONIQUE



### PROGRAMMATION ELEMENTAIRE D'UNE INTERRUPTION

Il existe dans le 80C196KC un registre interne 16 bits nommé SP pour **Stack pointer (pointeur de pile)**. Il est utilisé automatiquement par le micro-contrôleur pour adresser une zone RAM nécessaire à la sauvegarde de PC (compteur de programme) lors d'une interruption. Il convient donc d'initialiser SP avant toute autorisation d'interruption. Exemple: si la zone RAM affectée à la pile réside de 8000<sub>H</sub> à 8100<sub>H</sub> on peut écrire LD SP,#8100<sub>H</sub>. Ce qui charge l'opérande 8100<sub>H</sub> dans SP. SP ne sauve que des mots de 16 bits, l'initialisation doit donc être faite sur une adresse paire. SP fonctionne par auto-décrémentation lors de l'empilage, il faut toujours l'initialiser à l'adresse haute de la pile et non à l'adresse basse.

Exemple de programme avec interruption

- programme principal en 2080H
- programme d'interruption du port série en 6000H

**affectation globale :**

HPILE EQU 8100<sub>H</sub> ; affecter une étiquette à l'adresse haute de la pile  
(le sommet de la pile se nomme HPILE et se situe à l'adresse 8100<sub>H</sub>, c'est à partir de cette adresse que sera sauvegarder le contexte au moment de la prise en compte de l'interruption)

IX EQU 42<sub>H</sub> ; Créer un registre IX à l'adresse 42<sub>H</sub>  
(un registre 16 bits est créé à l'adresse 42<sub>H</sub> il se nomme IX)

**programme principal** : (ce programme débute à l'adresse 2080<sub>H</sub>)

2080<sub>H</sub> LD SP,#HPILE ; initialisation du pointeur de pile SP  
(le pointeur de pile SP est positionné à la valeur HPILE soit à 8100<sub>H</sub>, ce pointeur vise une zone mémoire allant de 8100<sub>H</sub> à 8000<sub>H</sub>, c'est dans cette zone que seront sauvegardés tous les éléments du programme en cours)

LD IX,#6000<sub>H</sub> ; initialisation du vecteur d'interruption associée à  
ST IX,200C<sub>H</sub> ; l'adresse de la routine d'interruption du port série  
(le port série bénéficie de l'interruption n°6, l'adresse du programme d'interruption correspondant doit se situer en 200C<sub>H</sub>, deux instructions sont nécessaires pour transférer la valeur 6000<sub>H</sub> (adresse de début du programme d'interruption) à cet emplacement)

LDB INT\_MASK,#01000000<sub>B</sub>; démasque uniquement SERIAL PORT  
(on autorise seulement la prise en compte d'une interruption émise par la circuiterie du port série en plaçant un 1 dans le bit 6 du registre INT\_MASK et un 0 dans tous les autres)

EI ; démasque les interruptions  
(on autorise la prise en compte globale des interruptions ; l'instruction EI vient positionner le bit de masquage général I du registre PSW (PSW.9) à 1 validant ainsi les interruptions)

suite du programme principal ;  
(le programme continue à la suite)

...

**sous programme d'interruption** : (ce programme débute à l'adresse 6000<sub>H</sub>)

6000<sub>H</sub> PUSHA ; empilage PSW, IMASK1 et INT\_MASK puis I=0  
(sauvegarde du mot d'état PSW et des masque d'interruption IMASK1 et INT\_MASK puis interdiction globale de la prise en compte d'une autre interruption)

programme d'interruption ;  
(le programme d'interruption continue à la suite )

...

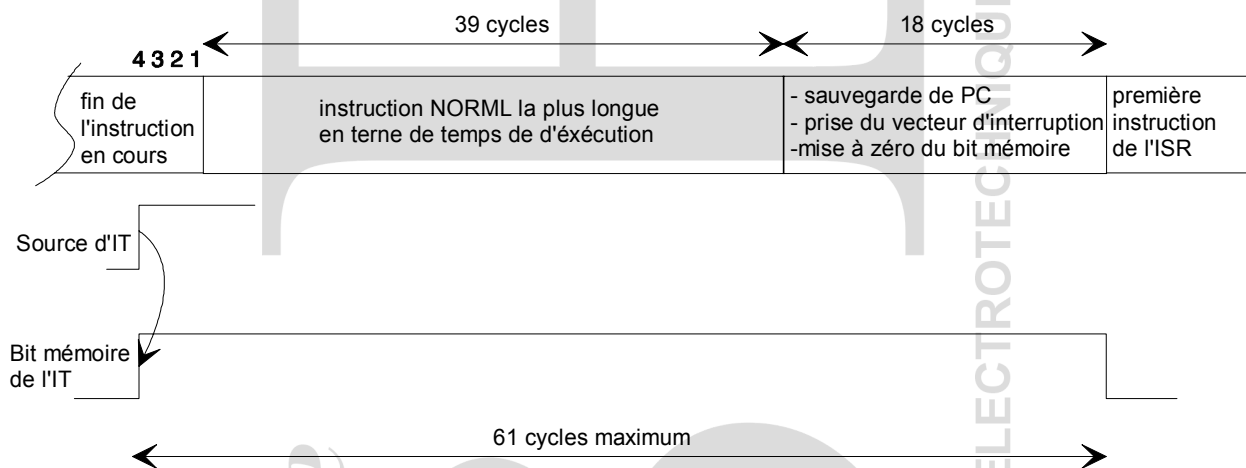
POPA ; restitution de PSW, IMASK1 et INT\_MASK  
(restitution du mot d'état PSW et des masque d'interruption IMASK1 et INT\_MASK sauvegardé avant l'exécution du programme d'interruption)

RET ; restitution de l'état de PC avant interruption  
(retour au programme principal à l'endroit où il avait été quitté)

### SOURCE ET VECTEUR D'INTERRUPTION

NUMERO	SOURCE D'INTERRUPTION	VECTEUR	PRIORITE
INT 15	NMI	203EH	15
INT 14	HSI FIFO FULL	203CH	14
INT 13	EXTINT PIN	203AH	13
INT 12	TIMER 2 OVERFLOW	2038H	12
INT 11	TIMER 2 CAPTURE	2036H	11
INT 10	HSI FIFO 4	2034H	10
INT 9	RI FLAG	2032H	9
INT 8	TI FLAG	2030H	8
SPECIALE	UNIMPLEMENTED OPCODE	2012H	N/A
SPECIALE	SOFTWARE TRAP	2010H	N/A
INT 7	EXTINT	200EH	7
INT 6	SERIAL PORT	200CH	6
INT 5	SOFTWARE TIMER	200AH	5
INT 4	HSI_0 PIN	2008H	4
INT 3	HIGH SPED OUTPUT	2006H	3
INT 2	HSI DATA AVAILABLE	2004H	2
INT 1	A/D CONVERSION COMPLETE	2002H	1
INT 0	TIMER OVERFLOW	2000H	0

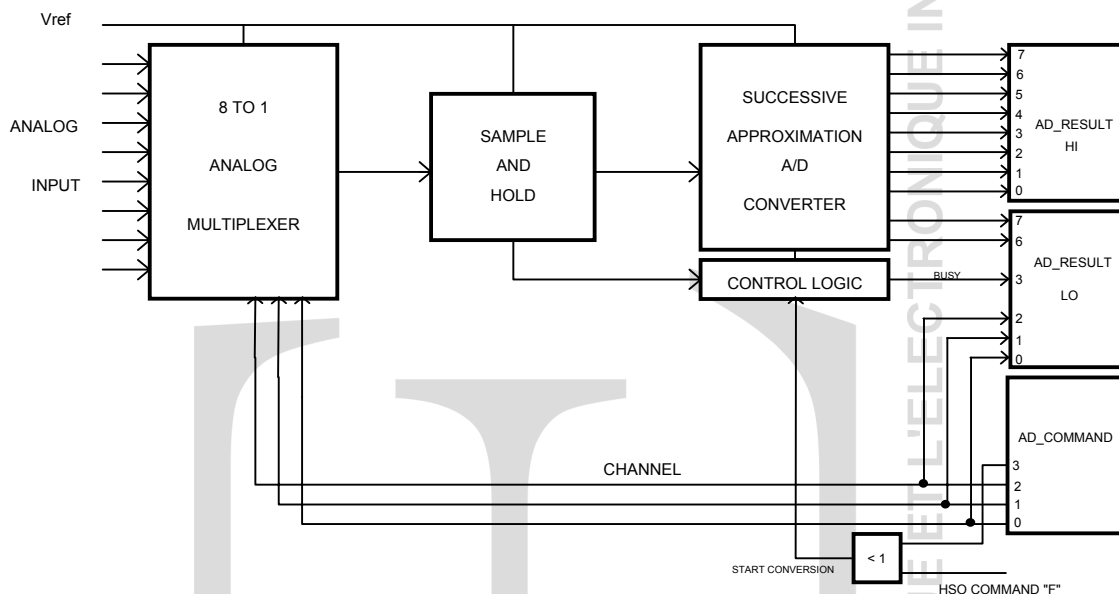
### TEMPS DE REPONSE DU SYSTEME D'INTERRUPTION



Le schéma précédent montre que l'intervalle de temps maximum séparant l'instant d'apparition d'un événement et l'instant d'entrée dans l'ISR correspondant peut prendre au maximum 61 state times (coups d'horloge). Pour un quartz à 16 MHz cela donne:  $125\text{ns} \times 61 = 7,6\mu\text{S}$  de retard.

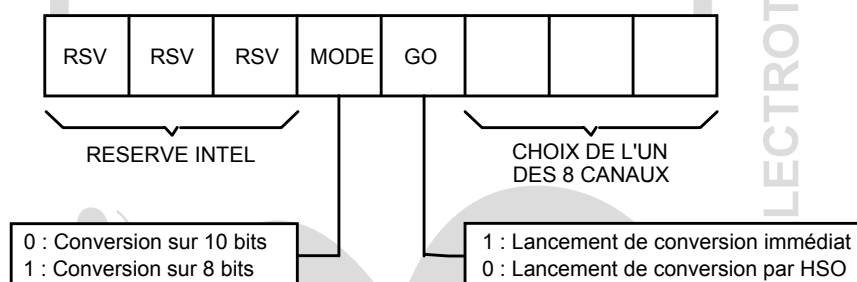
## 7. Ports d'entrées de conversion analogique / numérique

Le circuit de Conversion Analogique Numérique du micro-contrôleur 80C196KC a 8 entrées analogiques multiplexées (on peut donc convertir une seule voie à la fois). Ces 8 entrées utilisent les mêmes broches que le port 0. Un circuit échantillonneur bloqueur (sample and hold) permet le maintien de l'entrée analogique pendant la phase de conversion. Le résultat de la conversion sur 8 ou 10 bits est placé dans le registre AD\_RESULT. Le lancement d'une conversion est fait soit par le registre AD\_COMMAND soit par une HSO (high speed output).



Nous avons une résolution de 19,53 mV ( 8 bits ) ou de 4,88 mV ( 10 bits ) pour une dynamique d'entrée de 5 V. Pour une fréquence du quartz de 16 MHz une conversion sur 8 bits peut être réalisée en 9,8125µs. Pour lancer la conversion, choisir la vitesse du convertisseur et pour connaître le résultat, nous avons besoin de 5 bits du registre AD\_COMMAND.

### REGISTRE DE COMMANDE AD\_COMMAND ( 02<sub>H</sub> de SFRs )



### REGISTRE DE CONTROLE D'ENTREE/SORTIE IOC2 ( 0B<sub>H</sub> des SFRs ):

Deux bits IOC2.3 et IOC2.4 contrôlent la vitesse du convertisseur analogique numérique.

La vitesse du convertisseur dans le mode 10 bits peut être ajustée en utilisant ou non un diviseur d'horloge. Caractéristique pour une conversion 10 bits

Pour IOC2.3 = 0

Clock préscaler on : IOC2.4 = 0 158 states times soit 19,7 µs à 16 MHz

Clock préscaler off : IOC2.4 = 1 91 states times soit 11,3 µs à 16 MHz

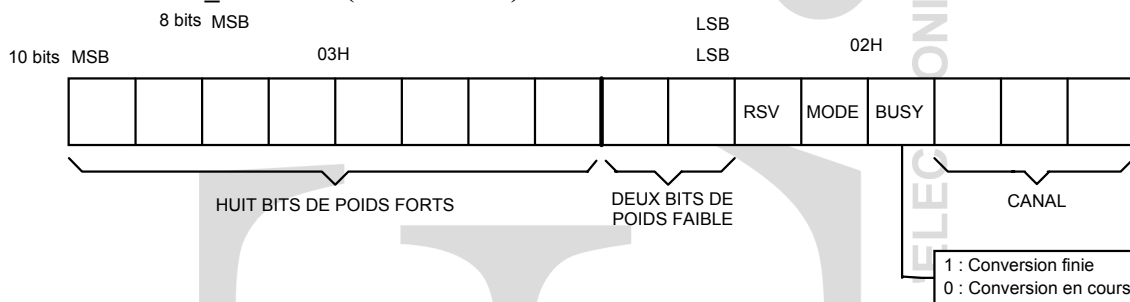
Si IOC2.3 = 1 et IOC2.4 = X (indifférent) la vitesse de conversion est alors définie par le contenu d'un registre SFR supplémentaire : AD\_TIME (non étudié ici).

### REGISTRE AD\_TIME (03H de HWINDOW 1)

L'utilisateur peut choisir le temps d'échantillonnage (les 3 MSB de AD\_TIME) et le temps de conversion (les 5 LSB de AD\_TIME). Si le temps d'échantillonnage est trop court, la capacité d'échantillonnage ne sera pas chargée correctement ; s'il est trop long, l'entrée analogique pourrait changer et introduire des erreurs. Le temps de conversion doit être assez long pour que le comparateur puisse basculer et assez court pour que la capacité d'échantillonnage ne se décharge pas. En somme, le choix n'est pas quelconque et un résultat correct ne sera atteint qu'avec une valeur particulière. Pour une conversion 10 bits (AD\_TIME) = 0C7<sub>H</sub> soit avec un quartz 16 MHz un temps de conversion de 13,3125 µs. Pour une conversion 8 bits (AD\_TIME) = 0A6<sub>H</sub> soit avec un quartz 16 MHz un temps de conversion de 9,8125 µs.

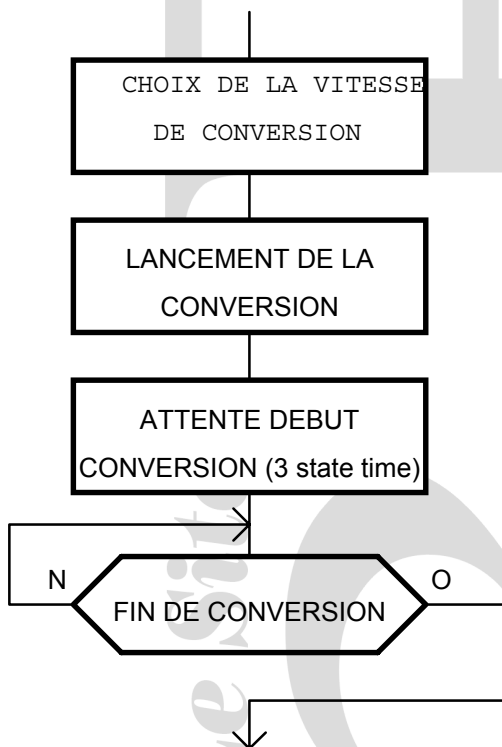
Remarque : Lorsque l'on choisit une CAN en 8 bits AD\_TIME doit être absolument utilisé.

### REGISTRE AD\_RESULT ( 02H et 03H ):



Pour savoir si la conversion est terminée, il suffit de tester le bit 3 de AD\_RESULT.

### ORGANIGRAMME CONVERSION



### PROGRAMME D'UNE CONVERSION

```
LDB  IOC2,#00000000B
(conversion lente avec pré-diviseur )
LDB  AD_COMMAND,#00001001
(lancement immédiat d'une conversion sur le canal 1
en mode 10 bits)
NOP
NOP
(laisser 3 cycles avant de tester NOP = 2 cycles)
LOOP:
BBC  AD_RESULT,3,LOOP
(boucler tant que le bit 3 du registre AD_RESULT
est à 0 (conversion en cours) lorsque le bit passe à 1
continuer (conversion terminée))
LD   AD_SAVE,AD_RESULT
(charger le résultat de la conversion dans AD_SAVE
qui aura été défini par une instruction EQU avant)
```

### LES INTERRUPTIONS:

Si on lance la conversion à l'aide des sorties rapides HSO, on peut générer une interruption dont le vecteur d'interruption est Software\_Timer (200A<sub>H</sub>).

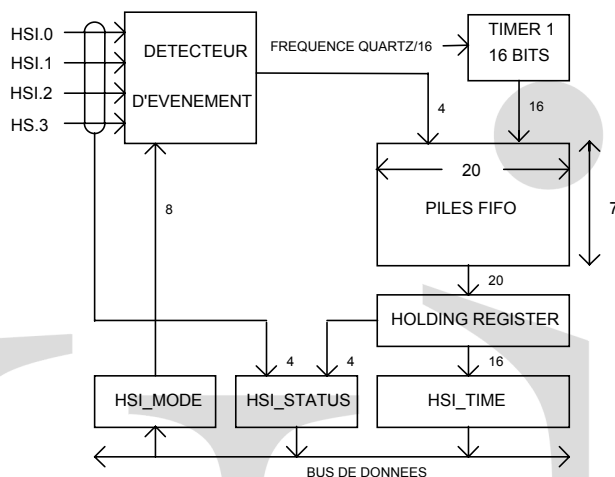
A la fin de la conversion, on peut générer une interruption afin de lancer un programme spécifique qui va rechercher le résultat de la conversion. Le vecteur d'interruption de A/D\_CONVERSION COMPLET est à l'adresse 2002<sub>H</sub>.

## Travail personnel

### 1. Mise en œuvre du module d'entrée rapide

Ce sont des entrées qui, lorsqu'un événement arrive sur l'une d'elles, enregistrent automatiquement la date et l'état logique de celui-ci dans une pile du  $\mu C$ .

On peut enregistrer comme cela, 8 événements consécutifs avec leurs dates.



On pourra utiliser ces entrées pour mesurer la durée des impulsions:

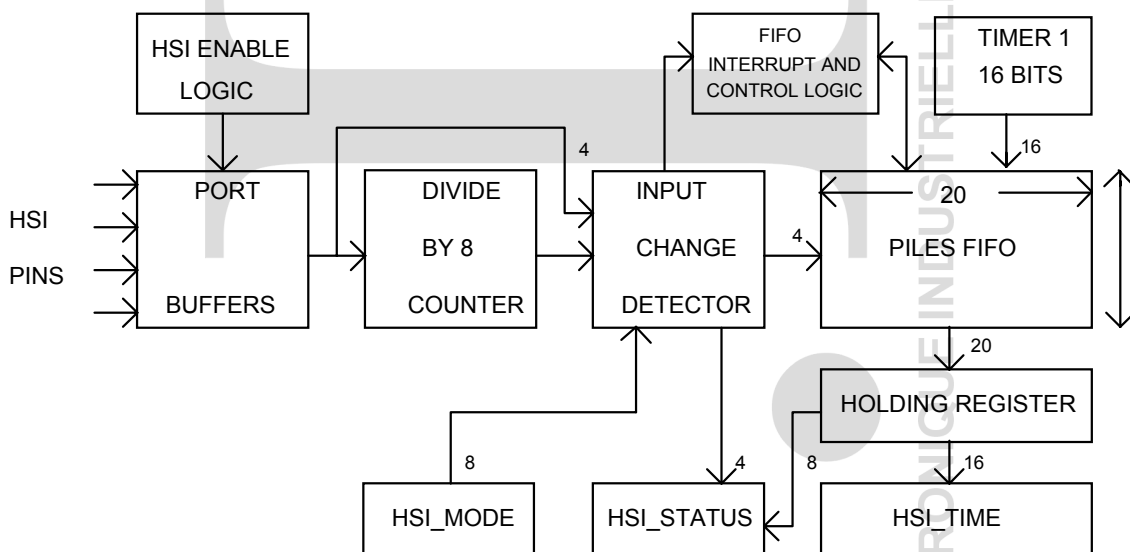
- Premier événement : Date D1 sur un front montant
- Deuxième événement : Date D2 sur un front descendant
- Durée de l'impulsion =  $D2 - D1$ .

De la même façon, on pourra mesurer la fréquence d'un signal :

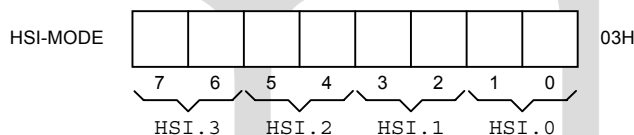
- Premier événement : Date D1 sur un front montant
- Deuxième événement : Date D2 sur un front montant
- Période du signal  $T = D2 - D1$  mise à l'échelle en seconde fréquence du signal  $F = 1/T$ .

On peut donc tout simplement connaître la vitesse de rotation d'un moteur par une mesure du nombre d'impulsion ou de l'écart entre les impulsions et faire son asservissement.

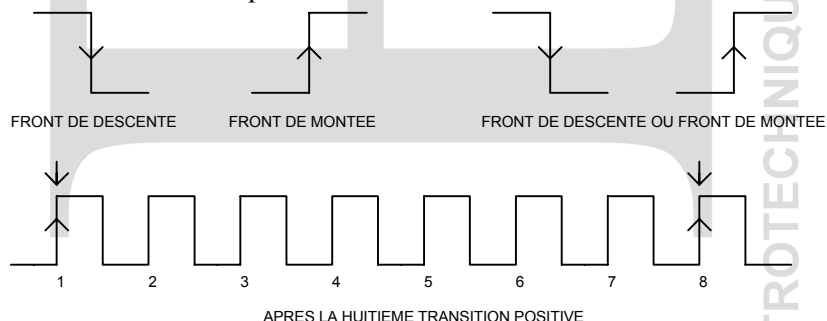
**SYNOPTIQUE:**



Au regard de ce synoptique, un certain nombre de registres sont utiles afin d'exploiter les HSI. Le **registre HSI\_MODE** (HWINDOW 0, adresse 03<sub>H</sub>) nous permet de choisir les transitions actives à enregistrer sur les 4 entrées rapides HSI.



On a quatre modes d'événements possibles:



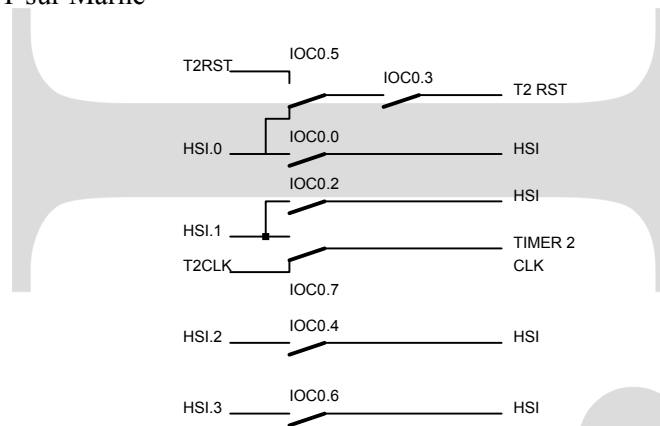
Suivant le contenu du registre HSI\_MODE, nous choisissons un des quatre modes ci-dessus pour chaque HSI.

BITS		MEMORISATION DE L'EVENEMENT
0	0	après huit transitions positives
0	1	après une transition positive
1	0	après une transition négative
1	1	après une transition positive ou négative

Le **registre IOC1** qui se trouve à l'adresse 16<sub>H</sub> de la HWINDOW 0 permet de valider les HSI.2 et HSI.3. : Pour valider le fonctionnement de HSI.2, il faut IOC1.4 = 0. Pour valider le fonctionnement de HSI.3, il faut IOC1.6 = 0

Le **registre IOC0** qui se trouve à l'adresse 15<sub>H</sub> de la HWINDOW 0 permet la sélection des entrées HSI.



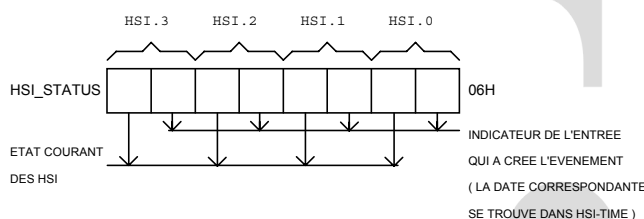


- Pour valider HSI.0, il faut mettre IOC0.0 = 1
  - Pour valider HSI.1, il faut mettre IOC0.2 = 1
  - Pour valider HSI.2, il faut mettre IOC0.4 = 1
  - Pour valider HSI.3, il faut mettre IOC0.6 = 1
- Par ailleurs, HSI.0 peut être utilisée pour mettre à zéro ( RESET ) le TIMER 2 et HSI.1 peut être utilisée comme horloge du TIMER 2.



Le **registre HSI\_STATUS** se trouve à l'adresse 06<sub>H</sub> de la fenêtre HWINDOW 0 en lecture. Deux informations sont disponibles dans ce registre :

- Etat courant des HSI.
- Le numéro de l'entrée qui a créé l'événement (la date étant dans HSI\_TIME).



Le **registre HSI\_TIME** se trouve à l'adresse 04<sub>H</sub> de la fenêtre HWINDOW 0 en lecture (registre 16 bits). Ce registre contient la valeur du compteur TIMER 1 (compteur s'incrémentant à chaque  $\mu$ s) pour le premier événement. Les dates des événements suivants (7) sont contenues dans la pile *FIFO* (premier entré, premier lu). La lecture de HSI\_TIME fait descendre automatiquement "d'un étage" les événements contenus dans la pile. Pour vider cette pile, il faudra donc faire 8 lectures de HSI\_TIME.

**ATTENTION :** Pour ne pas avoir des pertes d'informations entre la date de l'événement et l'entrée correspondante il faut: d'abord lire HSI\_STATUS puis lire HSI\_TIME.

**REMARQUE :** 8 cycles sont utiles pour prendre en compte un événement nouveau dans le HOLDING\_REGISTER. Pour effectuer la nouvelle lecture du registre HSI\_TIME, il faut attendre ce délai (Temps de gestion de FIFO).

Le **registre IOS1** se trouve à l'adresse 16<sub>H</sub> de la fenêtre HWINDOW 0 en lecture.

Deux indicateurs sont utilisés pour gérer les HSI:

- IOS1.6 = 1 Indique que la pile est pleine.
- IOS1.7 = 1 Indique qu'un événement est disponible dans HSI\_STATUS et HSI\_TIME (HSI\_TIME est chargé par la date relative à laquelle a eu lieu l'événement exprimé en  $\mu$ s).

**REMARQUE :** Avant d'utiliser les HSI, il faut initialiser la pile FIFO. En effet, à la mise sous tension du 80C196KC, le contenu de cette pile est aléatoire.

- Il faut lire 8 fois HSI\_TIME afin de vider la FIFO et le HSI\_HOLDING\_REGISTER.

#### REGLES D'UTILISATION DES HSI :

1. Vider la FIFO.
2. Initialisation : Validation des entrées rapides dans IOC0  
Validation HSI.2 et HSI.3 avec IOC1.4 et IOC1.6 = 0.  
Choix du mode dans HSI\_MODE.
3. Tester le bit IOS1.7 afin de savoir si le HOLDING\_REGISTER est chargé et peut être lu.
4. Lire HSI\_STATUS, le sauvegarder; Il indique l'entrée qui a généré événement.
5. Lire HSI\_TIME, son contenu sur 16 bits indique la date de événement généré par l'entrée détectée dans HSI\_STATUS.
6. Attendre 8 states time ( 8 tops d'horloge interne ) avant de faire une nouvelle lecture de HSI\_STATUS et HSI\_TIME ( 4 instructions NOP ).

#### UTILISATION DES HSI EN MODE INTERRUPTIBLE:

Quatre types d'interruptions peuvent être générés par les HSI.

- Si IOC1.7 = 0: une interruption sera générée lorsque le HOLDING\_REGISTER est chargé.
- Si IOC1.7 = 1: une interruption sera générée lorsque le contenu de la pile FIFO passe de l'état 2 places vide à l'état 1 place vide :

- Lorsque  $INT\_MASK1.2 = 1$ : une interruption sera générée dès qu'un 4ème événement arrivera dans la pile.
- L'entrée HSI.0 peut être utilisée pour générer une interruption sur chaque front montant.

### 1. Mise en œuvre du module d'entrée rapide (travail demandé)

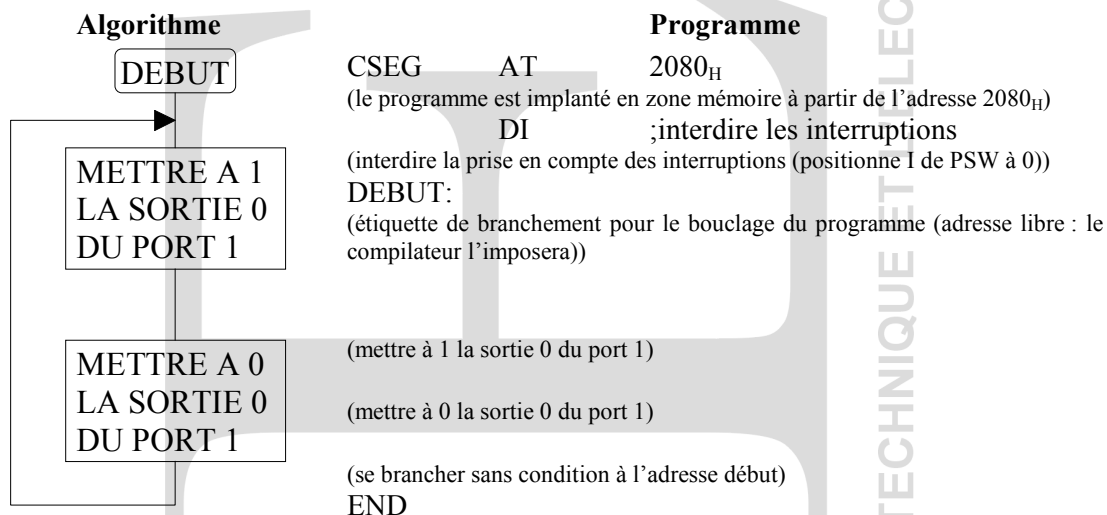
Rédiger un organigramme dans lequel vous devez mesurer la vitesse de rotation d'un moteur (vitesse de rotation comprise entre 10 et 100 tr/min) associé à un codeur d'impulsions de 256 points par tour.

- Le codeur est relié à l'entrée HSI.0
- On souhaite avoir le résultat en tr/min dans un mot de 16 bits

### 2. Ecriture d'un programme en assembleur à partir d'un organigramme

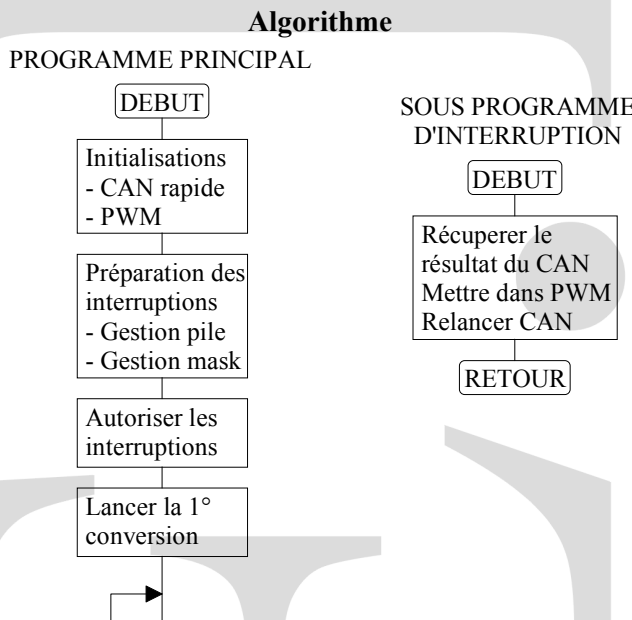
- Utilisation du jeu d'instruction joint en annexe
- Rédaction de programme commentée

Rédiger un programme en le commentant dans lequel vous devez mettre à 1 puis à 0 la sortie 0 du port 1 (P1.0) sans délais entre les deux opérations. Puis recommencer à l'infini (voir algorithme).



### 3. Ecriture d'un programme avec interruption en assembleur à partir d'un organigramme

Commenter le programme correspondant à l'algorithme ci-dessous



#### Affectations

RSEG	AT	30 <sub>H</sub> ; à partir de l'adresse 30 <sub>H</sub>
	DSB	9 ; réservation de 9 emplacements 8 bits
CSEG	AT	2000 <sub>H</sub> ; à partir de l'adresse 2000 <sub>H</sub>
	DCW	0 ; vecteur d'interruption non utilisé aux adresses 2000 <sub>H</sub> et 2001 <sub>H</sub>
	DCW	SOUS_PROG ; vecteur d'interruption correspondant à A/D CONVERSION COMPLETE aux adresses 2002 <sub>H</sub> et 2003 <sub>H</sub>

#### Programme principal

```

CSEG  AT  2080H
      DI
      LDB  IOC1,#00000001B
      LDB  IOC2,#00010000B
      LD   SP,100H
      LDB  INT_MASK,#00000010B
      EI
      LDB  AD_COMMAND,#00001000B
BOUCLE: BR  BOUCLE
  
```

#### Sous programme d'interruption

```

CSEG  AT  3000H ; à partir de l'adresse 3000H
SOUS_PROG:
      LDB  PWM0_CONTROL, AD_RESULT_HI
      LDB  AD_COMMAND,#00001000B
      RET
  
```

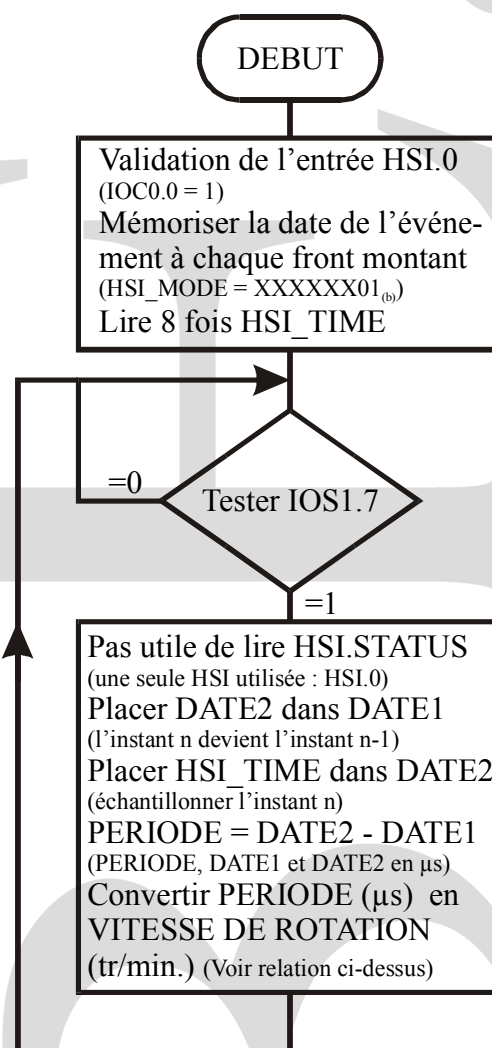
END

## Autocorrection

### 1. Mise en œuvre du module d'entrée rapide

- ✎ On va mesurer la période d'une impulsion (en  $\mu\text{s}$ ) issue du codeur incrémental, en échantillonnant les instants correspondants aux fronts montant sur l'entrée HSI.0 sur laquelle est raccordée ce codeur. Il suffira ensuite de convertir l'écart de temps compris entre deux échantillon successifs (en  $\mu\text{s}$ ) en une grandeur exprimant des  $\text{tr}/\text{min}$  :

$$n_{(\text{tr}/\text{min.})} = \frac{60}{256 \times \frac{\text{Période}_{(\mu\text{s})}}{10^{-6}}}$$

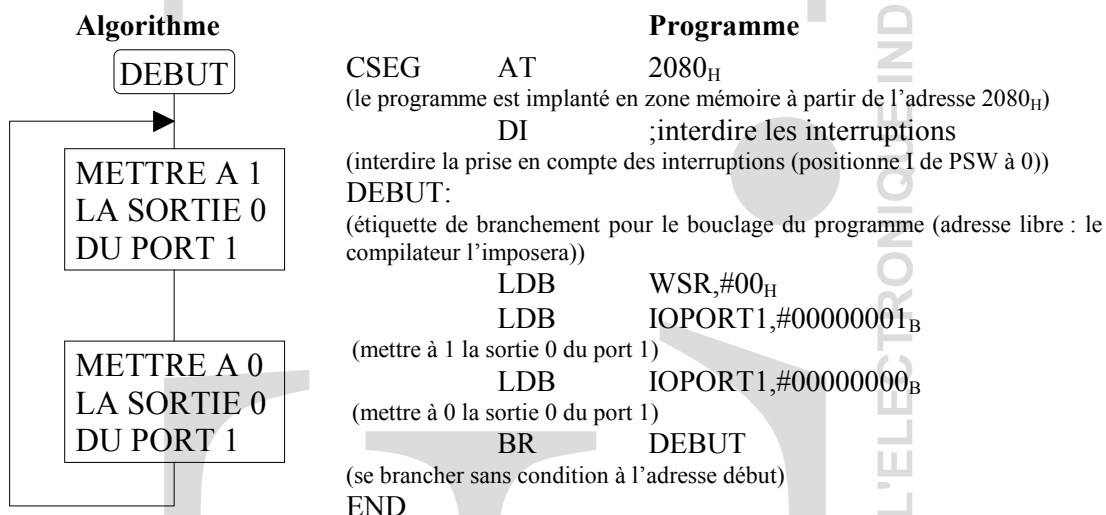


- ✎ Deux remarques s'imposent : Tout d'abord le premier calcul de la PERIODE donc de la VITESSE DE ROTATION est faux puisque DATE2 ne contient pas une valeur échantillonnée. Ensuite le TIMER 1 est codé en 16 bits, il compte donc de 0 à 65535 puis recommence à 0 ce qui entraîne un calcul de période faux qu'il faudrait éliminer. Un simple test vérifiant que DATE2 est supérieure à DATE1 suffit.

## 2. Ecriture d'un programme en assembleur à partir d'un organigramme

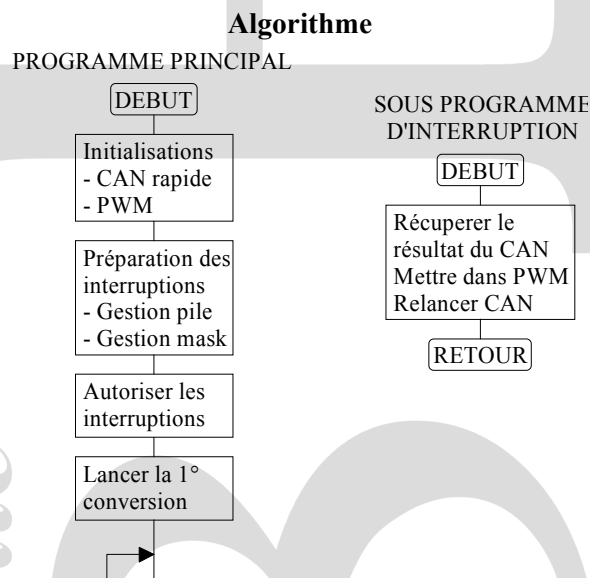
- Utilisation du jeu d'instruction joint en annexe
- Rédaction de programme commentée

Rédiger un programme en le commentant dans lequel vous devez mettre à 1 puis à 0 la sortie 0 du port 1 (P1.0) sans délais entre les deux opérations. Puis recommencer à l'infini (voir algorithme).



## 3. Ecriture d'un programme avec interruption en assembleur à partir d'un organigramme

Commenter le programme correspondant à l'algorithme ci-dessous



### Affectations

RSEG	AT	30 <sub>H</sub> ;	à partir de l'adresse 30 <sub>H</sub> :
	DSB	9 ;	réserve de 9 emplacements de 8 bits
CSEG	AT	2000 <sub>H</sub> ;	à partir de l'adresse 2000 <sub>H</sub> :
	DCW	0 ;	vecteur d'interruption non utilisé aux adresses 2000 <sub>H</sub> et 2001 <sub>H</sub>
	DCW	SOUS_PROG ;	vecteur d'interruption correspondant à A/D
			CONVERSION COMPLETE aux adresses 2002 <sub>H</sub> et 2003 <sub>H</sub>

Contenant l'adresse où se situe SOUS\_PROG

### Programme principal

CSEG AT 2080<sub>H</sub>

Le code est implanté à partir de l'adresse 2080<sub>H</sub> c'est là que débute le programme (voir zone adressable du µ-cont.)

DI

Interdiction des interruptions avant leur configuration

LDB IOC1,#00000001<sub>B</sub>

Sélection de la sortie PWM 0 avec connection sur la broche 39.

LDB IOC2,#00010000<sub>B</sub>

Initialisation de la vitesse de conversion analogique/numérique sans préscaler : durée de conversion = 11,3 µs.

Initialisation de la fréquence de la sortie PWM à 31,25 kHz

LD SP,100<sub>H</sub>

Affecter la valeur 100<sub>H</sub> à l'adresse haute de la pile nécessaire lors de la mise en œuvre des interruptions. (c'est à partir de cette adresse que sera sauvegarder le contexte au moment de la prise en compte de l'interruption).

LDB INT\_MASK,#00000010<sub>B</sub>

On démasque (c'est à dire que l'on autorise sa prise en compte) A/D\_DONE qui pourra à la fin d'un processus de conversion analogique/numérique interrompre le programme principal en cours d'exécution pour lui substituer le sous-programme d'interruption. Lorsque ce dernier est terminer le programme principal est repris là où il avait été interrompu.

EI

Validation des interruptions

LDB AD\_COMMAND,#00001000<sub>B</sub>

Lancement immédiat de la conversion analogique/numérique sur le canal 0 en mode 10 bits.

BOUCLE: BR BOUCLE

Le programme boucle sur lui même en attendant une interruption.

### Sous-programme d'interruption

CSEG AT 3000<sub>H</sub> ; à partir de l'adresse 3000<sub>H</sub>

Le sous-programme d'interruption est placé à partir de l'adresse 3000<sub>H</sub>

SOUS\_PROG:

L'étiquette du sous-programme d'interruption permet de placer la valeur 3000<sub>H</sub> dans les cases 2002<sub>H</sub> et 2003<sub>H</sub>

LDB PWM0\_CONTROL, AD\_RESULT\_HI

Prendre le résultat de la conversion et le placer dans le registre de contrôle de PWM 0.

LDB AD\_COMMAND,#00001000<sub>B</sub>

Lancement immédiat de la conversion analogique/numérique suivante sur le canal 0 en mode 10 bits.

RET

Retour au programme principal

END

Fin de la zone à compiler