

---

# Ingénierie de $\mu$ -contrôleurs, 4A MT

---

## 1<sup>ère</sup> séance : TP1 « mesure de la vitesse »

Intervenant : Nikolay Smagin

Responsable pédagogique : Christophe Delebarre



## But général de ces TPs

---

**Calculateur de la boîte de vitesse** : il gère le passage des vitesses et les différents modes d'utilisation, les commandes et consignes de l'utilisateur, l'affichage, l'auto-adaptativité des programmes (à la température de l'huile de boîte, au vieillissement, ...), ... C'est la programmation de ce calculateur qui servira d'illustration pour ce TP.

Le seul capteur que nous considérons est celui de sortie de boîte (proportionnel à la vitesse du véhicule), il fournit 20 « tops » par tour par usinage d'une roue sur l'arbre de sortie.

# Déroulement général de ces TPs

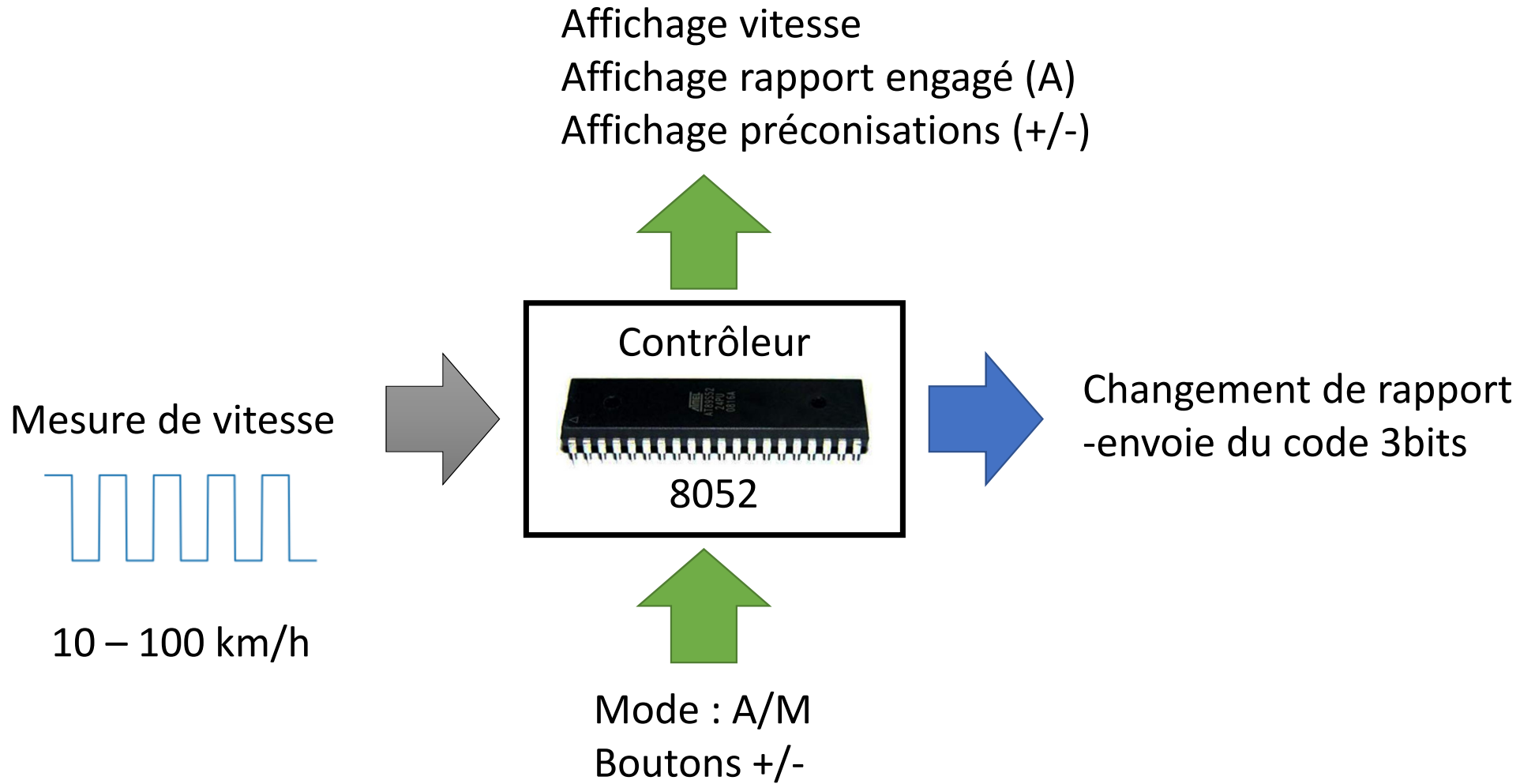
---

**TP1.** Récupération de l'information sur vitesse dans une plage 10 – 100 km/h (utilisation des temporisateurs) ; un critère de 5% sera admis sur la précision.

**TP2.** Affichage de la **vitesse** au port série du **mode** et du **rapport engagé**. Contrairement à l'énoncé joint ci-dessous, l'afficheur LCD ne sera pas utilisée car il n'est pas accessible par défaut sous Keil  $\mu$ Vision.

**TP3.** Gestion du passage automatique des rapports en fonction de la vitesse (mode auto).

# Contrôleur de la boîte de vitesses

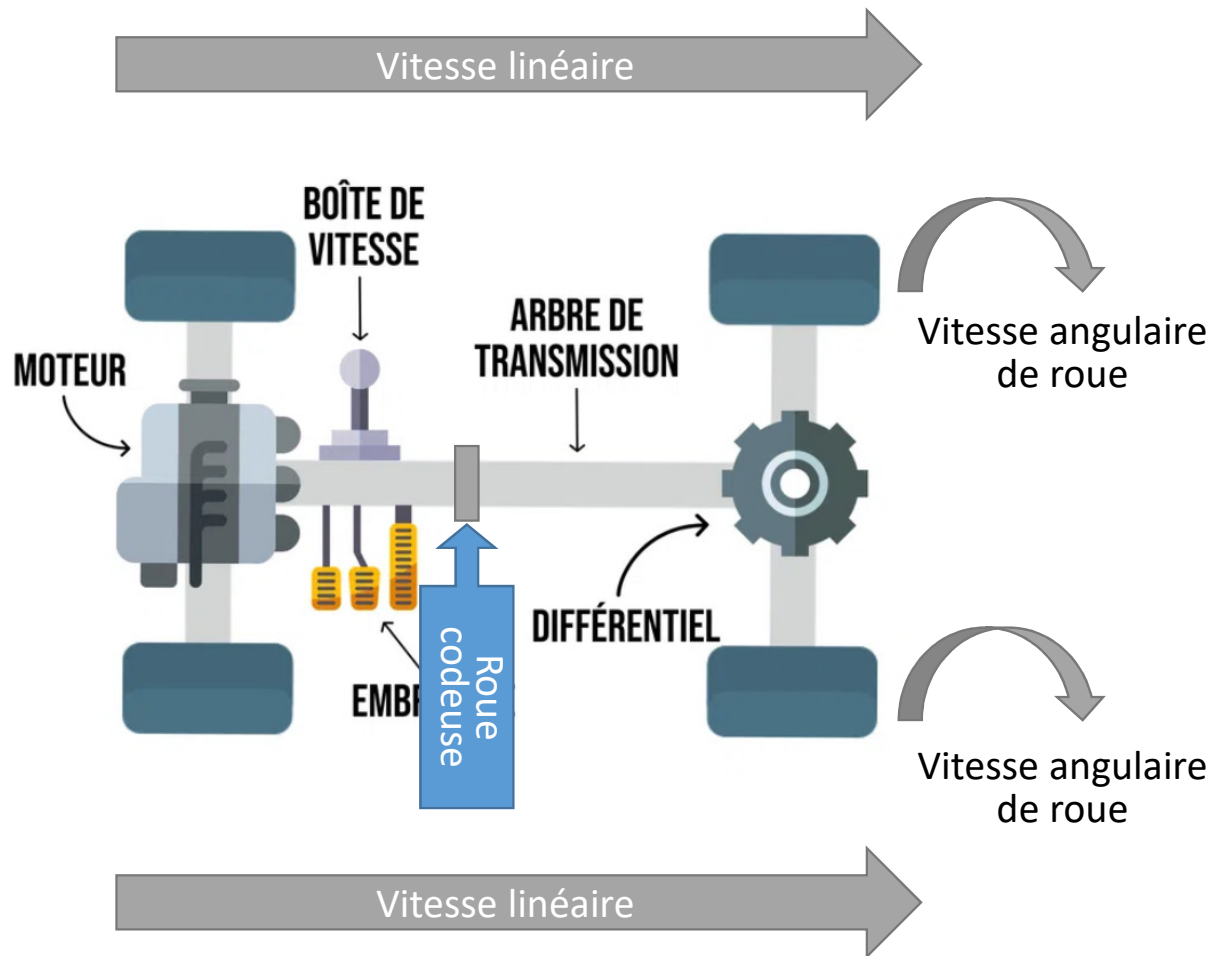


# Contrôleur de la boîte de vitesses

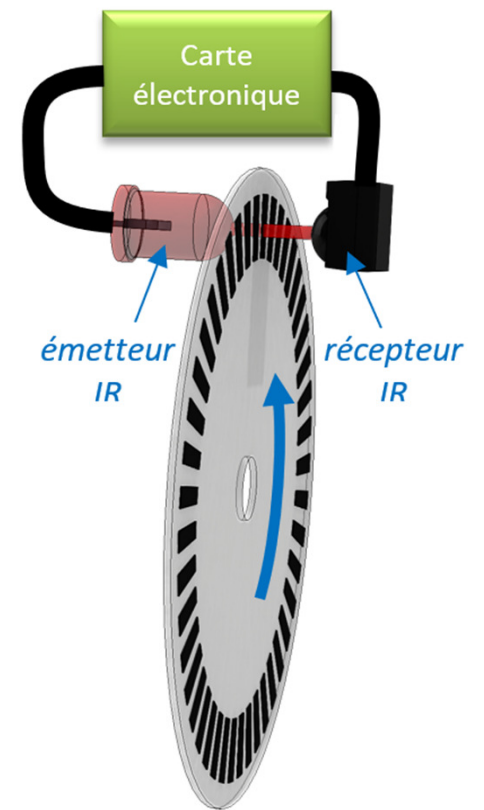
---

TP1.1 : Mesure de vitesse : dimensionnement

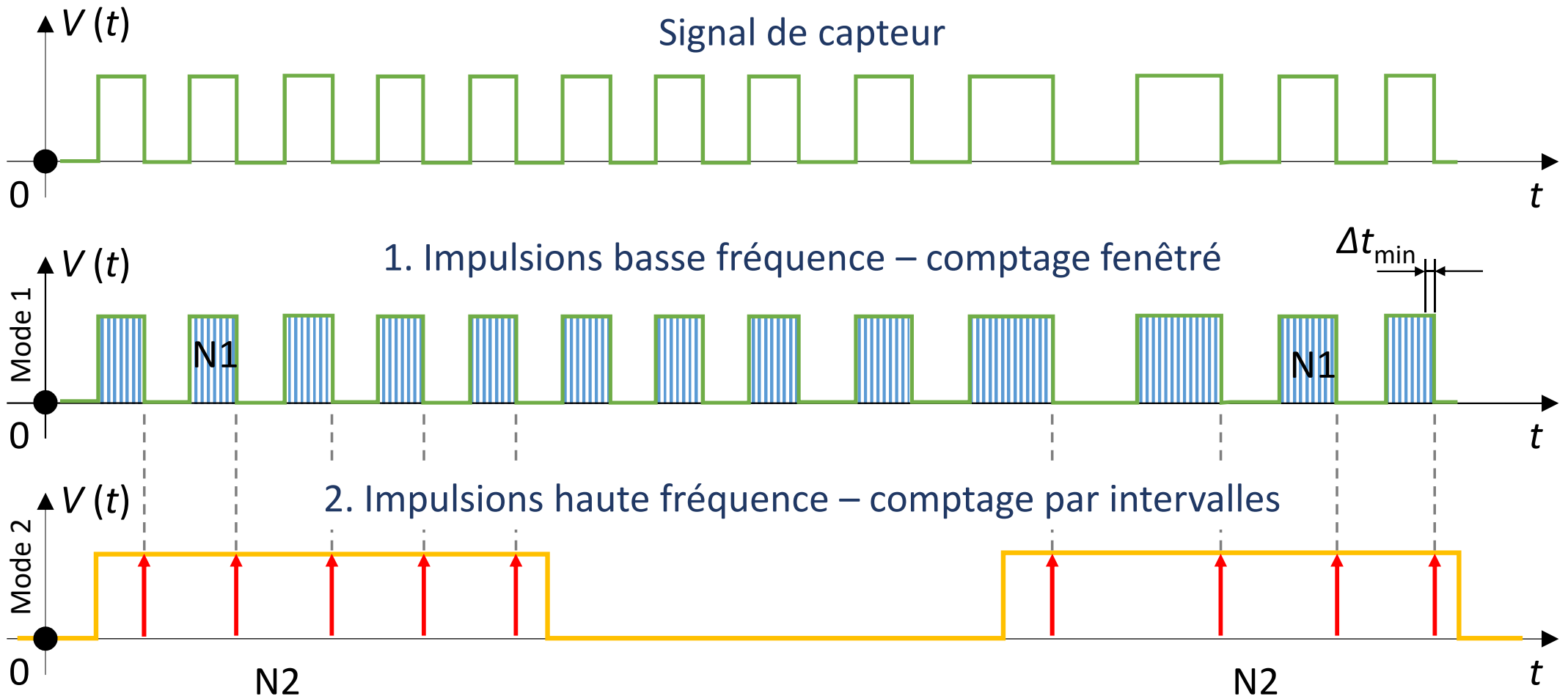
# Mesure de vitesse avec une roue codeuse



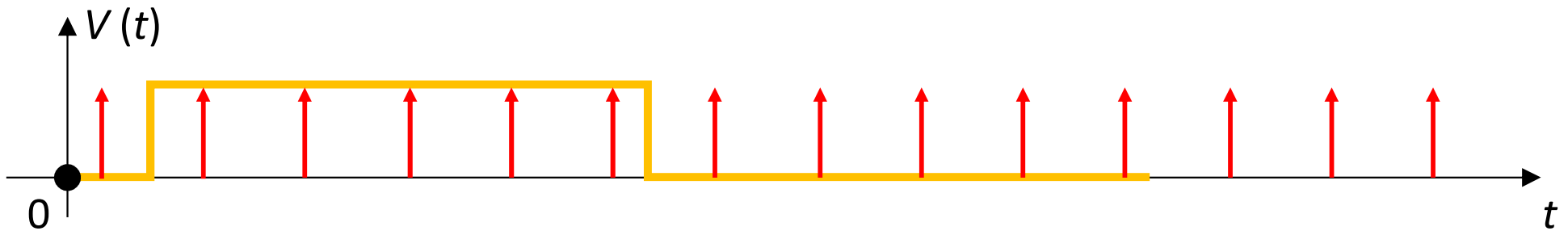
20 « tops » par tour de roue



# Mesure de fréquence : comptage des impulsions



# Estimation d'erreur



**Erreur de comptage :  $\pm 1$  impulsion**

$$v_1 = f(N_{impulsions})$$

$$v_2 = f(N_{impulsions} + 1)$$

$$erreur = \frac{\Delta v}{v_1} 100\% = \frac{v_2 - v_1}{v_1} 100\%$$

Function\_Mcycles sont aussi à prendre en compte

***erreur*  $\leq 5\%$**

# Estimation d'erreur

---

## Pour un comptage fenêtré

1. Il ne faut pas que le compteur déborde à  $v_{\min}$  (10 km/h)
2. Il faut qu'il ait assez de comptages pour  $v_{\max}$  afin que erreur < 5%

## ~~Pour un comptage par intervalles~~

- ~~1. Il faut qu'il ait assez de comptages pour  $v_{\max}$  (10 km/h) afin que erreur < 5%~~
- ~~2. Il ne faut pas que le compteur déborde à  $v_{\max}$  (100 km/h)~~
- ~~3. Il ne faut pas que  $T_{\text{intervalle}}$  soit trop long (latence < 500 ms)~~

# 1<sup>ère</sup> sous-partie TP1.1

---

- Dimensionner le projet : choisir le diamètre de roue R13 – R17 calculer les fréquences correspondantes à 10 – 100 km/h.
- Définir diagramme générale de programme, variables principales, etc.
- Comptage des impulsions représente une mesure de l'intervalle temporelle  $\Delta t$ , alors  $v \sim k / N_{\text{compt}}$ . Définir  $k$ . Conseil : Pour le calcul final la procédure DIV24\_16 sera nécessaire.
- Choisir le mode de mesure de vitesse (comptage fenêtré ou comptage sur une intervalle),  $f_{\text{OSL}} = 12 \text{ MHz}$ ;  $\Delta t_{\text{min}} = 1 \mu\text{s}$ .

# Contrôleur de la boîte de vitesses

---

TP1.2 : Mesure de vitesse (fréquence) avec les temporisateurs de AT89C51ED2

# Virtualisation de la tache

## Affichage vitesse

Affichage rapport engagé (A)  
Affichage préconisations (+/-)

En absence du matériel  
**TIMERO** fournira des signaux  
correspondant à 10 – 100 km/h

## Mesure de vitesse



**TIMERO**

10, 25, 45, 75, 95 km/h



Contrôleur



8052

**TIMER1**

Sera utilisé pour  
compter les impulsions

**Vitesse** -> **Fréquence** (à partir de dimensionnement)



**10, 25, 45, 75, 95 km/h**

**TIMERO**



**TIMER1**

Mesure cette **Fréquence** par comptage -> nombre **N**



**Conversion**

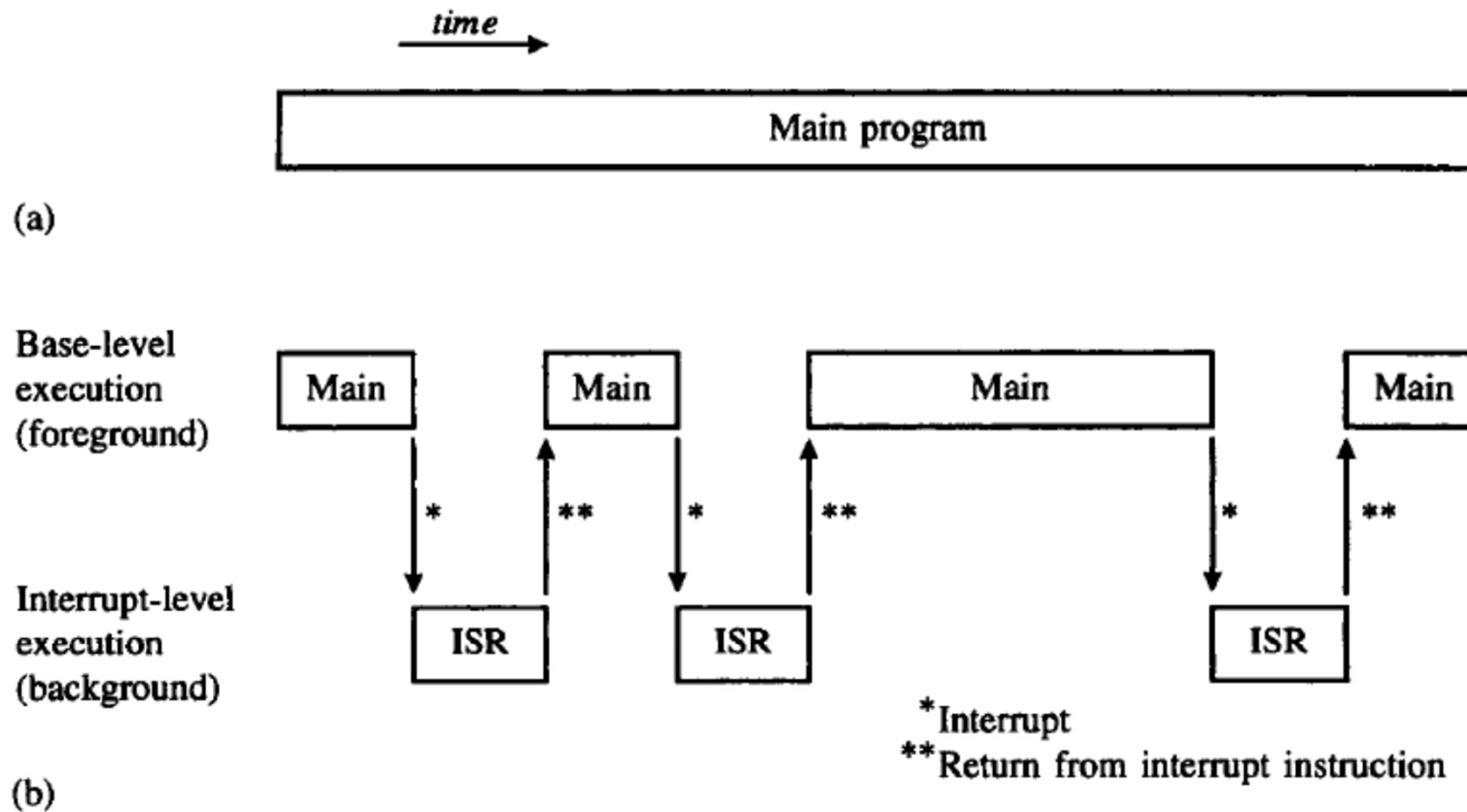
Avec la procédure DIV16\_16 (**k/N**) de votre énoncé, on retrouve la **Vitesse** ( $\pm 5\%$ )



**Affichage**

Par communication port série

# Fonctionnement par interruptions matérielles



**FIGURE 6-1**

Program execution with and without interrupts (a) Without interrupts (b) With interrupts

# Temporisateurs / Compteurs du 8051

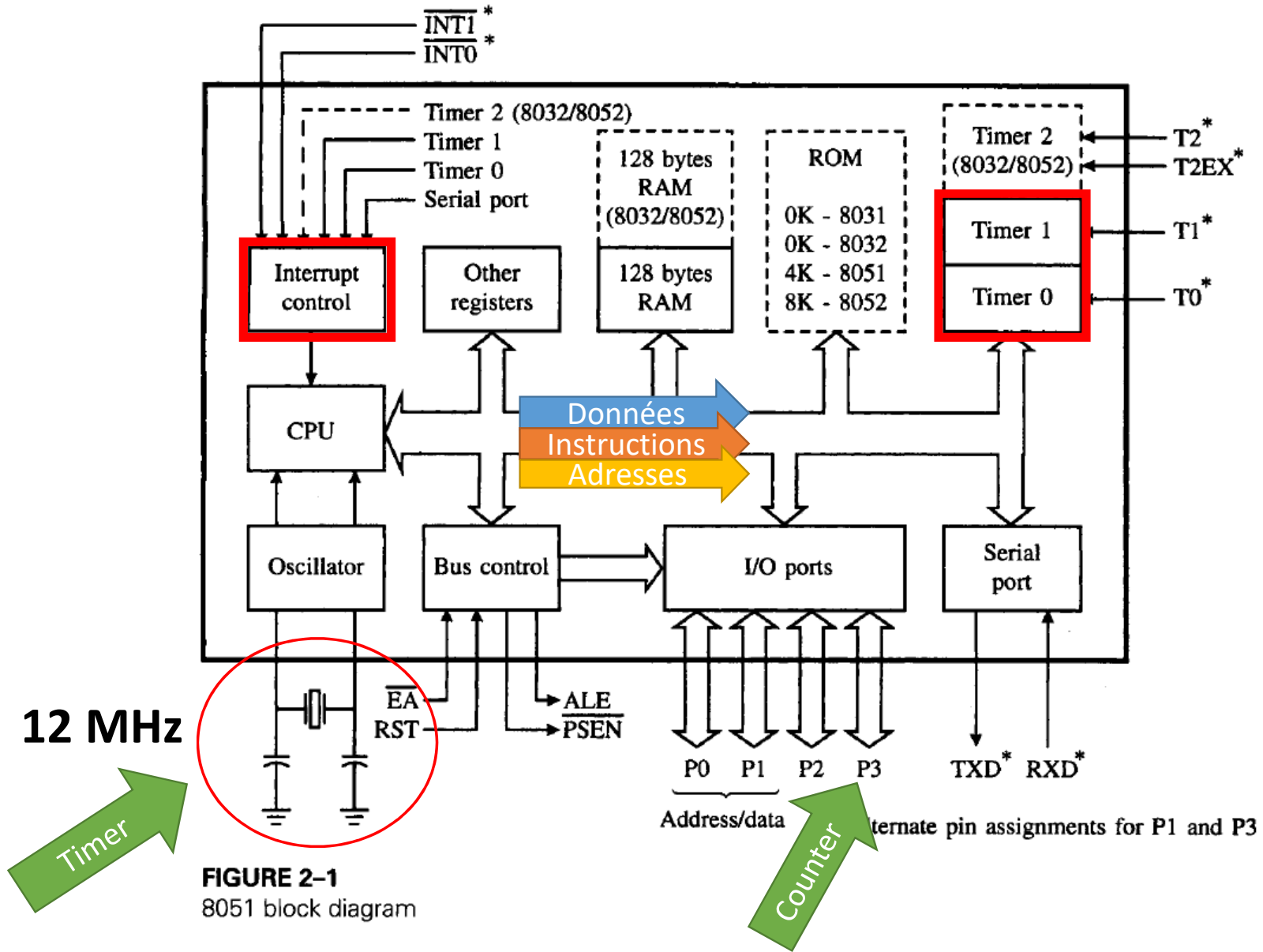
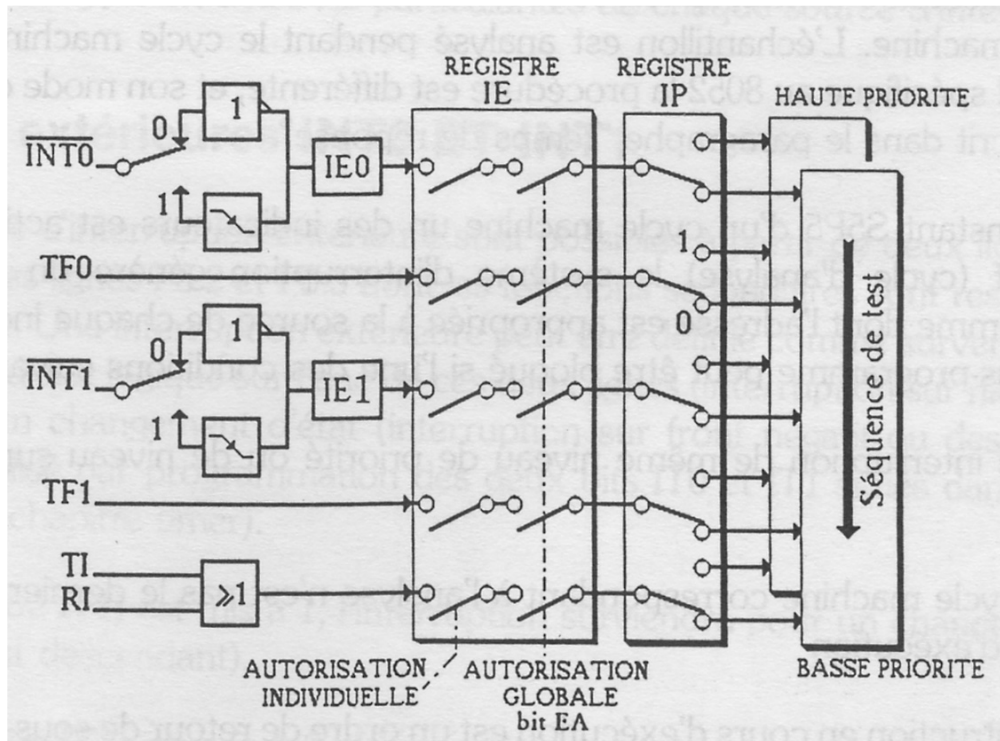


FIGURE 2-1  
8051 block diagram

# Fonctionnement par interruptions matérielles

Rappel : Ce sont les registres **IE** et **IP** qui sont responsables pour la configuration des interruptions



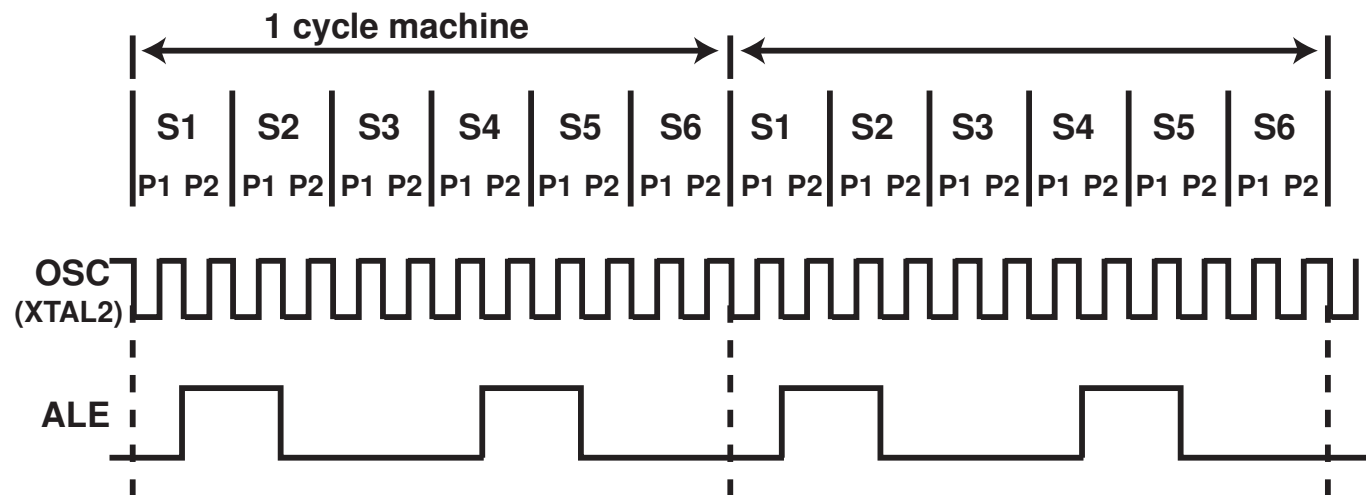
**TABLE 6-1**

IE (interrupt enable) register summary

Bit	Symbol	Bit Address	Description (1 = Enable, 0 = Disable)
IE.7	EA	AFH	Global enable/disable
IE.6	—	AEH	Undefined
IE.5	ET2	ADH	Enable Timer 2 interrupt (8052)
IE.4	ES	ACH	Enable serial port interrupt
IE.3	ET1	ABH	Enable Timer 1 interrupt
IE.2	EX1	AAH	Enable external 1 interrupt
IE.1	ET0	A9H	Enable Timer 0 interrupt
IE.0	EX0	A8H	Enable external 0 interrupt

SETB EA  
 SETB IE.7  
 SETB 0AFH

# Temporisateur mesure les battement de l'oscillateur (quartz)

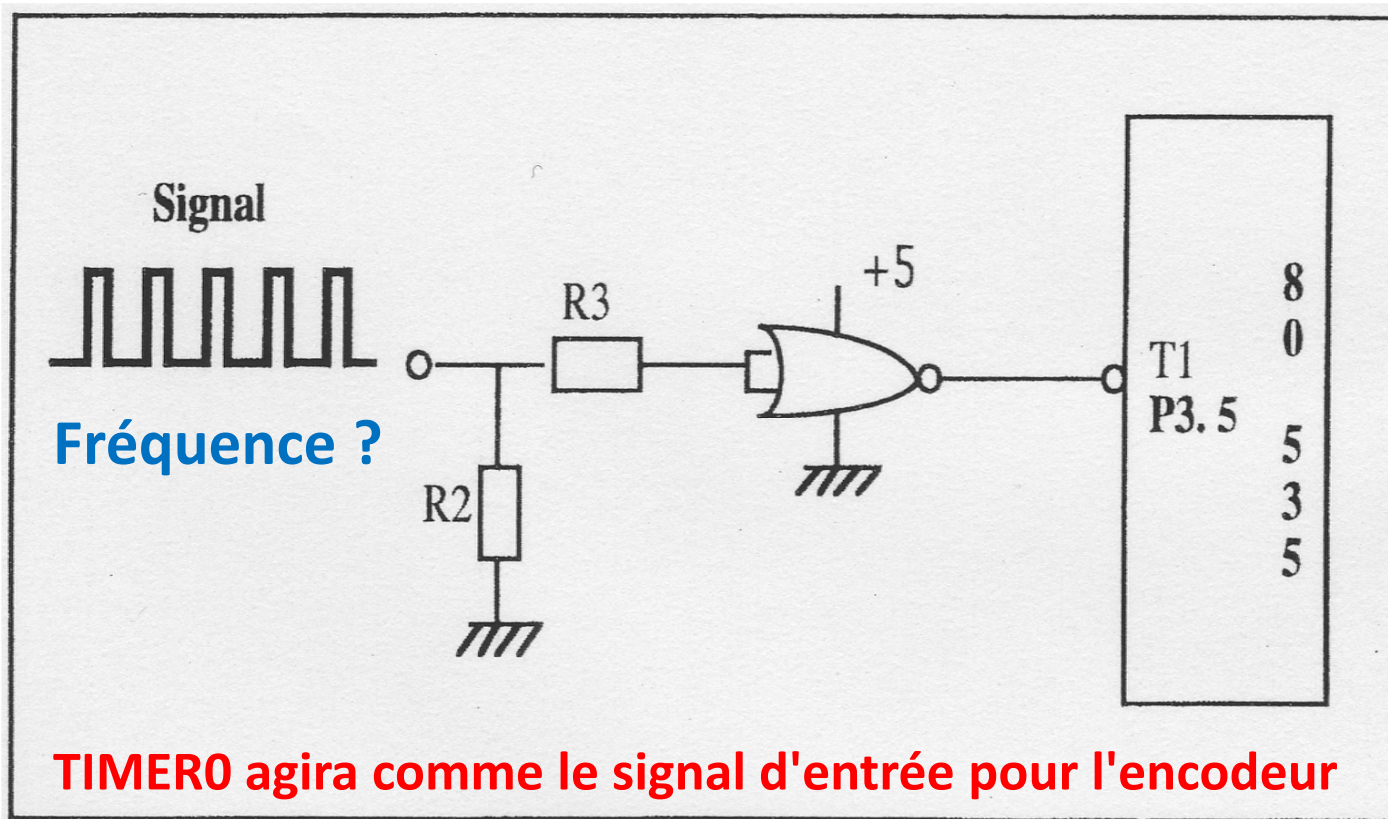
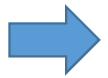


**Facteur 1/12**

➔ Signal de l'oscillateur interne divisé par 12. Il lui faut 1 cycles-machine pour incrémenter un temporisateur.

# Compteur mesure les battement de sur la broche T0 ou T1

Signal d'une roue codeuse  
optique pour mesurer la  
vitesse de rotation



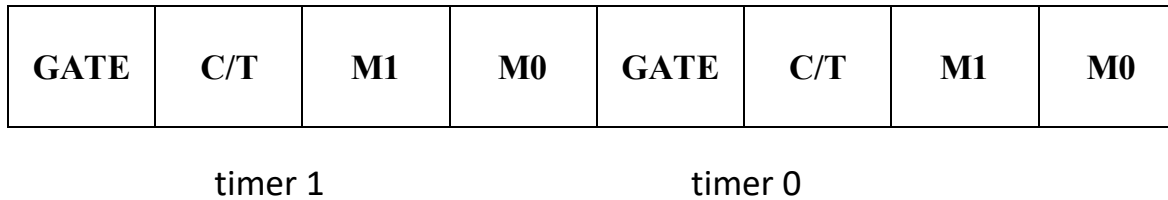
# Description des TIMERS

- ➔ Chaque Timer est composé d'un compteur 16 bits accessible sous la forme de 2 registres de 8 bits faisant partie du SFR
  - **TH0** et **TLO** pour le timer 0.
  - **TH1** et **TL1** pour le timer 1.
- ➔ Le registre **TMOD** permet le choix de la fonctions des modes.
- ➔ Le registre **TCON** concerne l'interruption, et le déclenchement du compteur.
- ➔ Chaque Timer possède 2 fonctions distinctes avec 4 modes de fonctionnement.

Byte address	Bit address	
FF		
F0	F7 F6 F5 F4 F3 F2 F1 F0	B
E0	E7 E6 E5 E4 E3 E2 E1 E0	ACC
D0	D7 D6 D5 D4 D3 D2 - D0	PSW
B8	- - - BC BB BA B9 B8	IP
B0	B7 B6 B5 B4 B3 B2 B1 B0	P3
A8	AF - - AC AB AA A9 A8	IE
A0	A7 A6 A5 A4 A3 A2 A1 A0	P2
99	not bit addressable	SBUF
98	9F 9E 9D 9C 9B 9A 99 98	SCON
90	97 96 95 94 93 92 91 90	P1
8D	not bit addressable	TH1
8C	not bit addressable	TH0
8B	not bit addressable	TL1
8A	not bit addressable	TLO
89	not bit addressable	TMOD
88	8F 8E 8D 8C 8B 8A 89 88	TCON
87	not bit addressable	PCON
83	not bit addressable	DPH
82	not bit addressable	DPL
81	not bit addressable	SP
80	87 86 85 84 83 82 81 80	P0

SPECIAL FUNCTION REGISTERS

# Description de TMOD (89H)

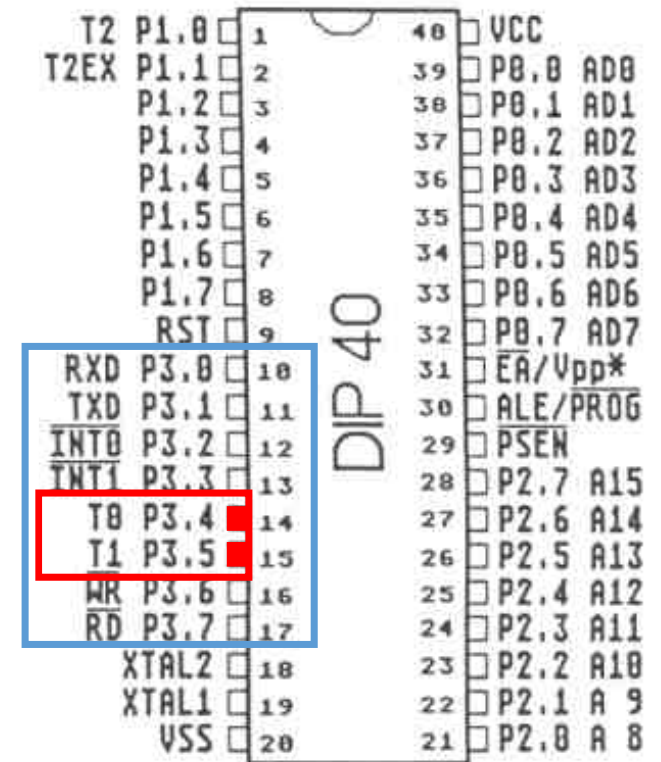


➔ **C/T**: permet la sélection de la fonction du TIMER

- C/T= 0 le timer est en fonction temporisateur : compteur incrémenté soit à partir du signal d'horloge, soit à travers un circuit de division de fréquence.
- C/T= 1 le timer est en fonction comptage : le compteur incrémenté par la détection d'événements extérieurs (changements d'état appliqués aux broches **T0=P3.4** ou **T1=P3.5**).

➔ **GATE** : permet de choisir le mode de déclenchement du compteur.

- GATE=0 : le timer x est validé si le bit « TRx » =1. (voir TCON)
- GATE=1 : le timer x est validé si le bit « TRx » =1 et que sa broche d'entrée « INTx » (P3.2 ou P3.3) est à l'état 1.



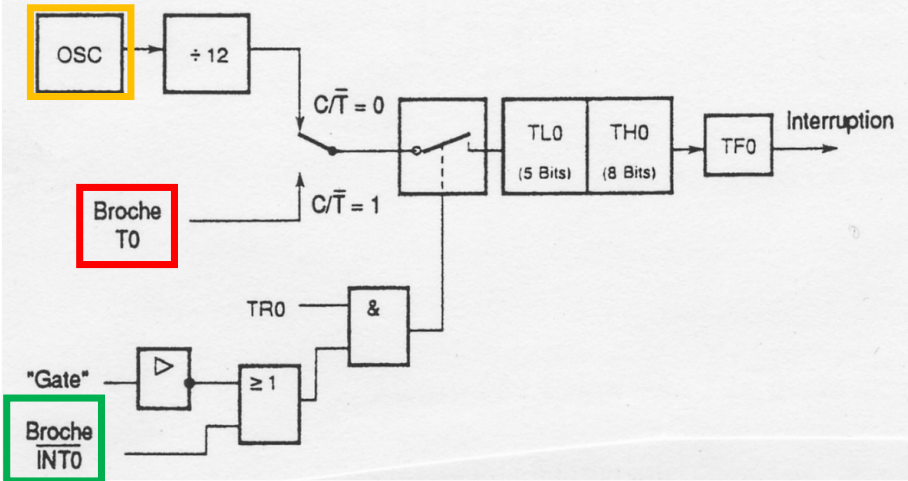
# Description de TCON (88H)

<b>TF1</b>	<b>TR1</b>	<b>TF0</b>	<b>TR0</b>	<b>IE1</b>	<b>IT1</b>	<b>IE0</b>	<b>IT0</b>
------------	------------	------------	------------	------------	------------	------------	------------

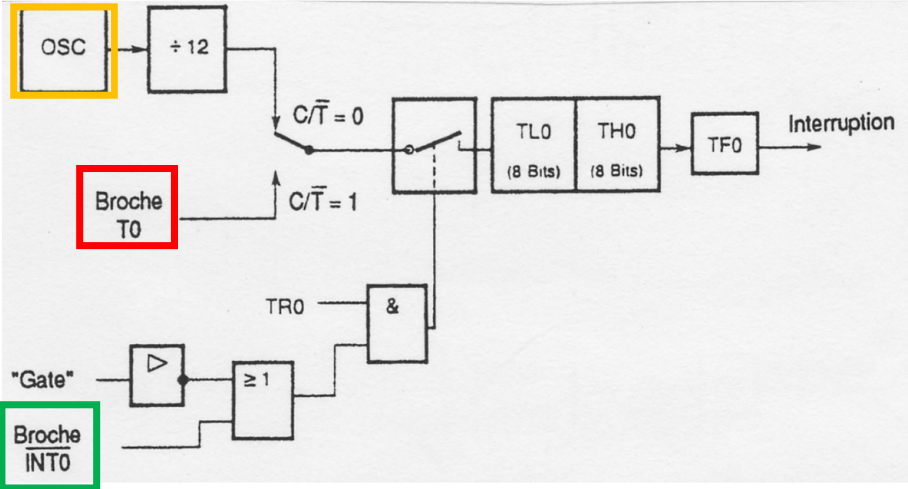
- ➔ **TFx** : Indicateur de débordement du TIMER x.  
Mis à 1 automatiquement lorsque le compteur repasse à 0.  
Si interruption correspondante autorisée, indicateur remis à 0 automatiquement lorsque le sous-programme (ISR) d'interruption est exécuté.
- ➔ **TRx** : Bit de déclenchement du TIMER x. (1 pour lancer, 0 pour arrêter).
- ➔ **IEx** : Indicateur de transition sur la source externe INTX (P3.2 ou P3.3).  
Cet indicateur est mis à 1 automatiquement lorsqu'une transition est détectée sur la borne INTx. Lors du service d'interruption, cet indicateur est remis à 0.
- ➔ **ITx** : Sélection du type d'événement devant déclencher l'interruption depuis la source INTx.  
ITx=0 interruption sur la détection d'un niveau 0 sur INTx.  
ITx=1 interruption sur la détection d'une transition négative.

# TMOD : 4 modes de temporisateurs/compteurs

■ Mode #00B



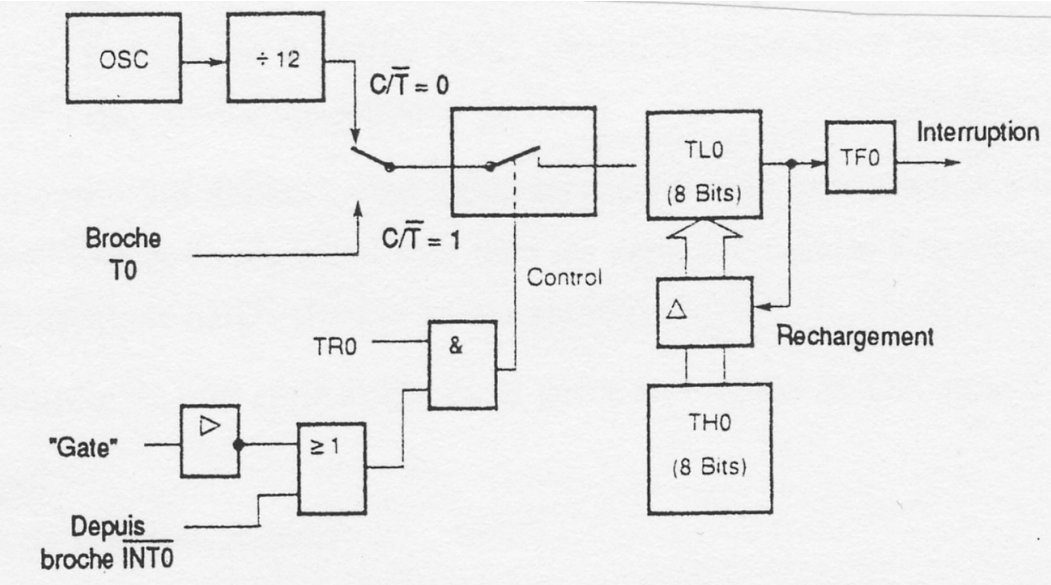
■ Mode #01B



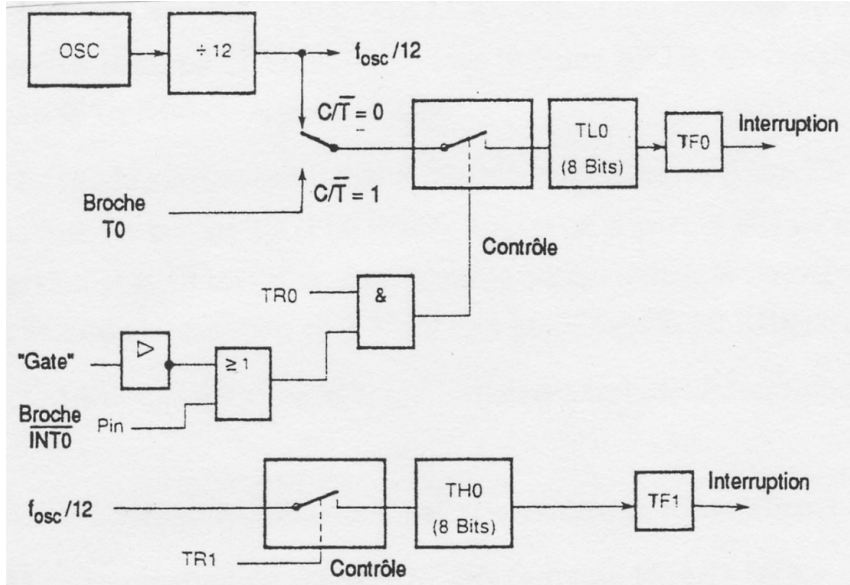
T2	P1.0	1	40	VCC
T2EX	P1.1	2	39	P0.8 AD0
	P1.2	3	38	P0.1 AD1
	P1.3	4	37	P0.2 AD2
	P1.4	5	36	P0.3 AD3
	P1.5	6	35	P0.4 AD4
	P1.6	7	34	P0.5 AD5
	P1.7	8	33	P0.6 AD6
	RST	9	32	P0.7 AD7
			31	EA/Vpp*
RXD	P3.8	10	30	ALE/PROG
TXD	P3.1	11	29	PSEN
INT0	P3.2	12	28	P2.7 A15
INT1	P3.3	13	27	P2.6 A14
T0	P3.4	14	26	P2.5 A13
T1	P3.5	15	25	P2.4 A12
WR	P3.6	16	24	P2.3 A11
RD	P3.7	17	23	P2.2 A10
XTAL2	18	22	21	P2.1 A9
XTAL1	19			P2.0 A8
VSS	20			

# TMOD : 4 modes de temporisateurs/compteurs

■ Mode #10B



■ Mode #11B

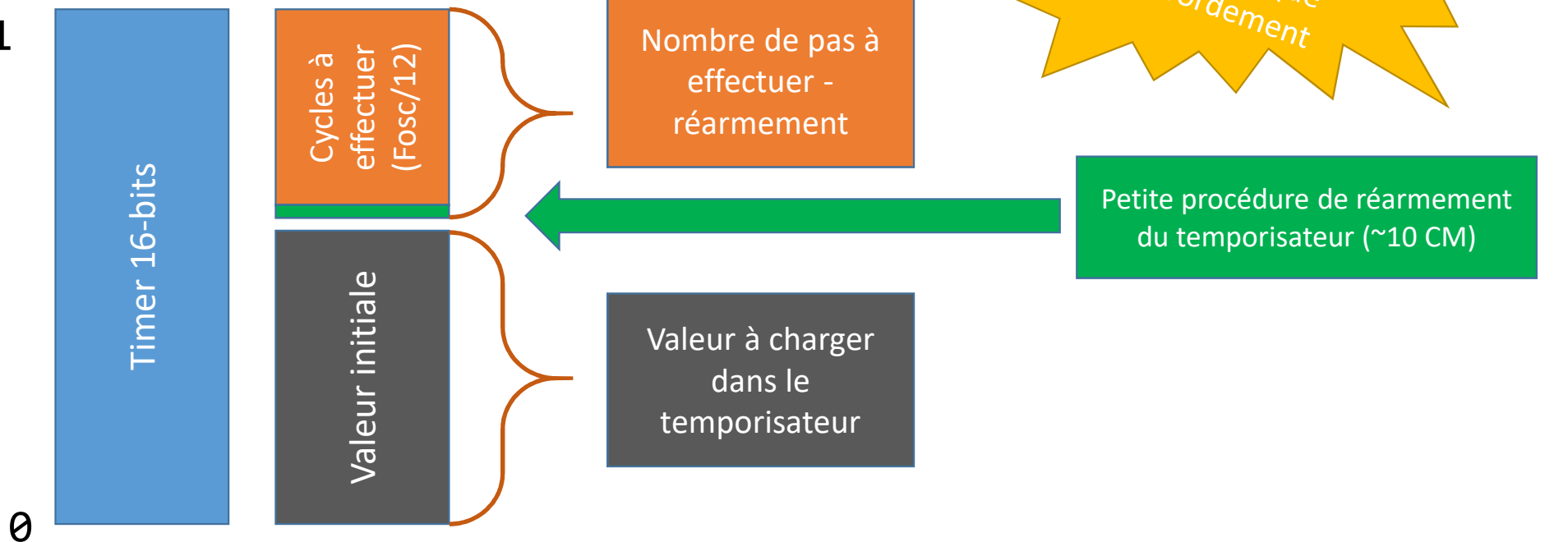


# Comment régler le timing (16 bits) ?

Value = (65535 - Count) + Function\_Mcycles + 1

Function\_Mcycles = 13 MC (machine cycles)

$2^{16} - 1$



# Réglage de temporisateur

---

*Calculation of Timer 0 reload value needed to achieve timer delay of 20 ms. Oscillator frequency is 11.0592 MHz.*

*Delay Value = Timer Delay / Timer Clock Cycle Duration*

$$= \frac{20 \times 10^{-3}}{12 \times 11.0592 \times 10^6}$$

*= 36864 (must be rounded to the nearest integer)*

*Timer Reload Value = Maximum Register Count - Delay Value*

$$= 65535 - 36864$$

$$= 28671$$

$$= 0x6FFF$$

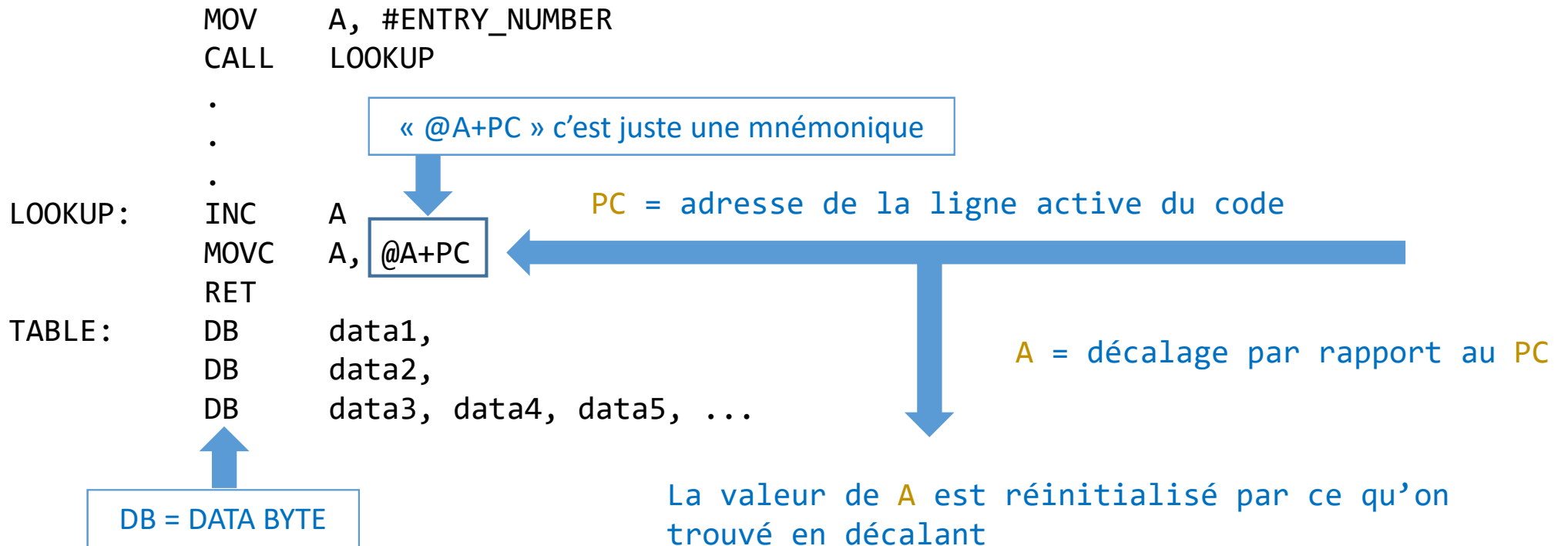
*so Timer 0 is loaded with:*

$$TH0 = 0x6F;$$

$$TL0 = 0xFF;$$

# Tableau de consultation (look-up table)

Ce code est équivalent à  $\{A = \text{TABLE}[A]\}$  en langage C (indexation)



# Exemple : Interaction de temporisateurs

The screenshot displays a microcontroller development environment with the following components:

- Registers:** A list of registers (r0-r7) and system registers (a-paw) with their current values.
- Logic Analyzer:** A timing diagram showing two signals, P1 (green) and P2 (red), over time. The P1 signal shows a series of pulses, and the P2 signal shows a high-frequency burst.
- Timer/Counter 0 and 1 Configuration:** Two dialog boxes showing the configuration for Timer/Counter 0 and Timer/Counter 1. Both are set to 1: 16 Bit Timer/Counter mode. Timer 0 has TCON: 0x10, TMOD: 0x11, TH0: 0xF8, TL0: 0x66, and is running. Timer 1 has TCON: 0x10, TMOD: 0x11, TH1: 0x3C, TL1: 0xB0, and is stopped.
- Code Editor:** A window titled "Notes\_pitch\_Major.a" containing assembly code for configuring and using the timers.

```
25 MOV IE, #10001010B ; |EA |---|---|ES |ET1|EX1|ET0|EX0| ; Timer 0 & 1 interrupts on
26 MOV R7, #0 ; use R7 as current frequency counter
27 MOV R6, #REPEAT ; use R6 as timeout counter
28 MOV PCNT, #PERSIST;
29
30 SETB TF1 ; force Timer 1 interrupt (notes)
31 SETB TF0 ; force Timer 0 interrupt (cadence)
32 SJMP $ ; ZzzZzzZzzZ time for a nap
33
34
35 ;*****
36 ; TIMER 0 INTERRUPT SERVICE ROUTINE (PITCH OF NOTES)
37 ;*****
```

Examinez le fichier [..]\[Fréquence variable]\Freq\_variable.uvproj

Simulation t1: 33.70893000 sec L:63 C:37 CAP NUM SCRL OVR R/W

## 2<sup>ème</sup> sous-partie du TP1.2

---

- Avec votre dimensionnement, régler la fréquence de TIMER0 pour imiter les signal du capteur correspondant à **10, 25, 45, 75, 95 km/h**.
- Effectuer la mesure temporelle des ces 5 fréquences avec TIMER1.

## 3<sup>ème</sup> sous-partie TP1.3

---

TP1.3 : Implémentation de fonctions

# Ressources utilisés

Registre	Fait quoi	Où
R7	Compteur de fréquence	TOISR
R0	Variable interne	DIV24_16
R1	Variable interne	DIV24_16
R2	Dividente octet bas	DIV24_16
R3	Dividente octet moyen	DIV24_16
R4	Dividente octet haut	DIV24_16
R5	Diviseur octet bas	DIV24_16
R6	Diviseur octet haut	DIV24_16
R7	Variable interne	DIV24_16
@R0	Adressage indirect port série	SPISR (pour le TP2)
Registre	Fait quoi	Où
P1.7	Etat de temporisateur 0 (TIMER0) – capteur de vitesse	TOISR
P3.3	Broche INT1 – pilotage de TIMER1	TOISR
P1.6	Procédure de division – observer le timing	DIV16_16

# SP = Stack Pointer

Un problème potentiel réside toutefois dans l'utilisation des registres dans les sous-routines. Au fur et à mesure que la hiérarchie des sous-routines se développe, il devient de plus en plus difficile de savoir quels registres sont affectés par les sous-routines. Une bonne pratique de programmation consiste donc à **enregistrer sur la pile les registres** qui sont modifiés par une sous-routine, puis à **les restaurer à la fin de la sous-routine**.

## Fonction

```
SQUARE:  PUSH    0F0H; préserver B
          MOV     0F0H, A
          MUL    AB
          POP    0F0H; restaurer B
          RET
```

PC est aussi stocké sur la pile lors des appels de fonctions (ACALL, LCALL) est retours (RET, RETI)

```
; A = SQUARE(A)
```

**SP** sert à **PC** pour lui indiquer où revenir après les sous-routines ou les interruptions

# SP = Stack Pointer

---

Souvent sont utilisées

FONCTION:

```
    push    00h    ;R0
    push    01h    ;R1
    push    ACC
    push    PSW
```

.... code

```
    pop     PSW
    pop     ACC
    pop     01h
    pop     00h
RET
```

# SP = Stack Pointer

```
MOV SP, #2FH
```

Pour redéfinir la pile ici et libérer les banques de registres (à l'initialisation)

```
SP = #07H
```

La valeur par défaut est #07H et la pile peut aller jusqu'à #7F

Byte address	Bit address	
7F		↓ données
		General purpose RAM
		↑ pile
30		
2F	7F 7E 7D 7C 7B 7A 79 78	
2E	77 76 75 74 73 72 71 70	
2D	6F 6E 6D 6C 6B 6A 69 68	
2C	67 66 65 64 63 62 61 60	
2B	5F 5E 5D 5C 5B 5A 59 58	
2A	57 56 55 54 53 52 51 50	
29	4F 4E 4D 4C 4B 4A 49 48	
28	47 46 45 44 43 42 41 40	
27	3F 3E 3D 3C 3B 3A 39 38	
26	37 36 35 34 33 32 31 30	
25	2F 2E 2D 2C 2B 2A 29 28	
24	27 26 25 24 23 22 21 20	
23	1F 1E 1D 1C 1B 1A 19 18	
22	17 16 15 14 13 12 11 10	
21	0F 0E 0D 0C 0B 0A 09 08	
20	07 06 05 04 03 02 01 00	
1F	Bank 3	
18	Bank 2	
17	Bank 1	
0F	Bank 1	
08	Default register bank for R0-R7	
07	Default register bank for R0-R7	
00	Default register bank for R0-R7	

RAM

Byte address	Bit address	
FF		
F0	F7 F6 F5 F4 F3 F2 F1 F0	B
E0	E7 E6 E5 E4 E3 E2 E1 E0	ACC
D0	D7 D6 D5 D4 D3 D2 - D0	PSW
B8	- - - BC BB BA B9 B8	IP
B0	B7 B6 B5 B4 B3 B2 B1 B0	P3
A8	AF - - AC AB AA A9 A8	IE
A0	A7 A6 A5 A4 A3 A2 A1 A0	P2
99	not bit addressable	
98	9F 9E 9D 9C 9B 9A 99 98	SBUF SCON
90	97 96 95 94 93 92 91 90	P1
8D	not bit addressable	
8C	not bit addressable	
8B	not bit addressable	
8A	not bit addressable	
89	not bit addressable	
88	8F 8E 8D 8C 8B 8A 89 88	TMOD TCON
87	not bit addressable	
83	not bit addressable	
82	not bit addressable	
81	not bit addressable	
80	87 86 85 84 83 82 81 80	DPH DPL SP P0

SPECIAL FUNCTION REGISTERS

La valeur de réinitialisation de 07H maintient la compatibilité avec le prédécesseur du 8051, le 8048, et a pour conséquence que la première écriture de pile stocke les données à l'emplacement 08H. Si le logiciel d'application ne réinitialise pas le SP, alors la banque de registres 1 (et peut-être 2 et 3) n'est pas disponible, puisque cette zone de RAM interne est la pile.

# Le compteur de programme et le pointeur de pile

The diagram illustrates the relationship between assembly code and hardware registers. On the left, a vertical stack of blue boxes represents 'Lignes de code' (code lines). A green arrow labeled 'Program counter' points downwards through the stack, indicating the current instruction being executed. Two red arrows labeled 'Boucle pr.' (loop) point to specific lines. In the center, a grey circle labeled 'Assemblage' (assembly) is connected to four arrows: 'Instructions' (orange), 'Adresses' (yellow), 'Donnés' (blue), and 'Registres - mémoire' (green). Below this is a photograph of a robotic arm assembly line with the text 'Langage d'Assembleur (« Assembly »)'. To the right, two screenshots from a debugger are shown. The 'Registers' window shows the 'PC \$' register highlighted in red with the value 'C:0x0003'. The 'Disassembly' window shows the instruction 'LJMP EX0ISR' at address 'C:0x0003' highlighted in yellow. A 'Memory 2' window shows the address 'D:0x81' and data values '09 00 00' and '00 00 00'.

**PC = Program counter** (espace de codes); **SP = Stack Pointer** (espace de données)

# Exemple : le compteur de programme

The screenshot shows a debugger interface with two main panes: 'Registers' on the left and 'Disassembly' on the right. In the 'Registers' pane, the 'PC \$' register is highlighted with a red box and contains the value 'C:0x0003'. In the 'Disassembly' pane, the instruction at address 'C:0x0003' is highlighted with a yellow box and contains the text '02005B L JMP EX0ISR (C:005B)'. A 'Memory 2' window is also visible, showing memory addresses D:0x81 and D:0xA5. At the bottom, a green banner contains the text: 'Examinez le fichier [..]\[Stack Pointer]\View\_PC\_SP.uvproj'.

Register	Value
r0	0x00
r1	0x00
r2	0x01
r3	0x00
b	0x01
sp	0x09
sp_max	0x09
dptr	0x0000
PC \$	C:0x0003
states	90
sec	0.00009000

```
8: L JMP EX0ISR
9:
10:
11: ORG 50h
C:0x0003 02005B L JMP EX0ISR (C:005B)
C:0x0006 00 NOP
C:0x0007 00 NOP
C:0x0008 00 NOP
C:0x0009 00 NOP
C:0x000A 00 NOP
C:0x000B 00 NOP
C:0x000C 00 NOP
C:0x000D 00 NOP
C:0x000E 00 NOP
```

TP3\_7seg\_display.a

5  
6  
7 ORG 0003h

Examinez le fichier [..]\[Stack Pointer]\View\_PC\_SP.uvproj

**PC** = **Program counter** (espace de codes) ; **SP** = **Stack Pointer** (espace de données)

## 3<sup>ème</sup> sous-partie du TP1.3

---

- Implémenter la fonction `div24_16` de l'énoncé en faisant attention à sauvegarder le registres sur la pile
- Sauvegarder la valeur de vitesse dans une variable 8 bits. Elle sera utilisée par la suite.

TP1 : Récupération de l'information sur vitesse dans une plage 10 – 100 km/h

---

*Bon courage !*

**Au supplément du compte-rendu pensez à m'envoyer les projets entiers Keil  $\mu$ Vision : le dossier entier compressé incluant les fichiers \*.uvproj et \*.a**

## 2<sup>ème</sup> sous-partie du TP1.2

On utilisera ce tableau pour les seuils de passages de rapports pour TP2–TP3

Nom	Vitesses	Valeur (km/h)
	passages de rapports +	
S01	0 → 1	9
S12	1 → 2	19
S23	2 → 3	39
S34	3 → 4	69
S45	4 → 5	89
	passages de rapport -	
S54	5 → 4	76
S43	4 → 3	56
S32	3 → 2	30
S21	2 → 1	16
S10	1 → 0	10