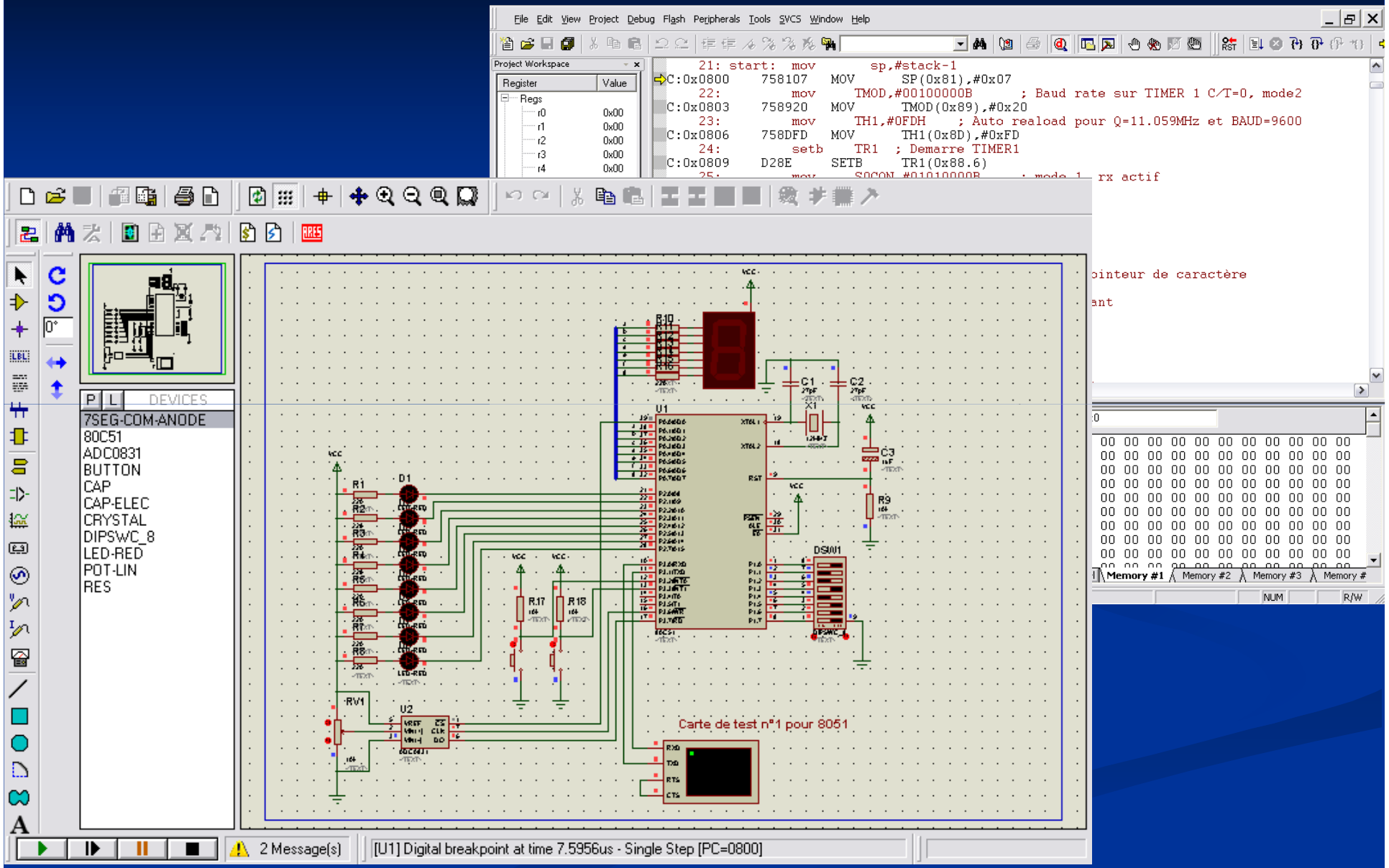
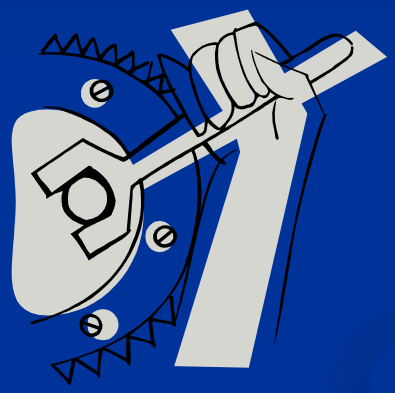
uVISION et PROTEUS-ISIS-VSM

Simulation-exemple



TP N°1



Les modes d’adressages, Précisions et compléments

MODE D 'ADRESSAGE (1)

Immédiat

MOV A, # 0x7F

Base	Suffix	Legal Characters	Examples
Hexadecimal	H, h	0 – 9, A – F, a – f	0x1234 0x99 1234H 0A0F0h 0FFh
Decimal	D, d	0 – 9	1234 65590d 20d 123
Octal	O, o, Q, q	0 – 7	177o 25q 123o 177777q
Binary	B, b	0 and 1	10011111b 101010101b

Base numérique

Symbole de l'adressage immédiat

A ?

Exemples:

Valeur numérique

```
MY_VAL     EQU     50H
```

Directive assembleur

```
MOV        A,#0E0h
MOV        DPTR,#0x8000
ANL        A,#128
XRL        A,#0FFh
MOV        R5,#MY_VAL
```

Assembleur

Direct

MOV A, 0x35

? A

Adresse Effective = la valeur
Adresse possible 0 .. 255
Permet l 'accès à la RAM basse et aux SFRs

2F	34H
4A	35H
55	36H

Exemples:

```
VALUE:      DS       1
IO_PORT2    DATA     0A0H
VALUE2      DATA     20H
```

Directives assembleurs

Adresse mémoire: 0 .. 255

```
MOV        A,IO_PORT2
ADD        A,VALUE
MOV        VALUE2,A
MOV        R1,#VALUE
```

Assembleur

- **Direct par registre**

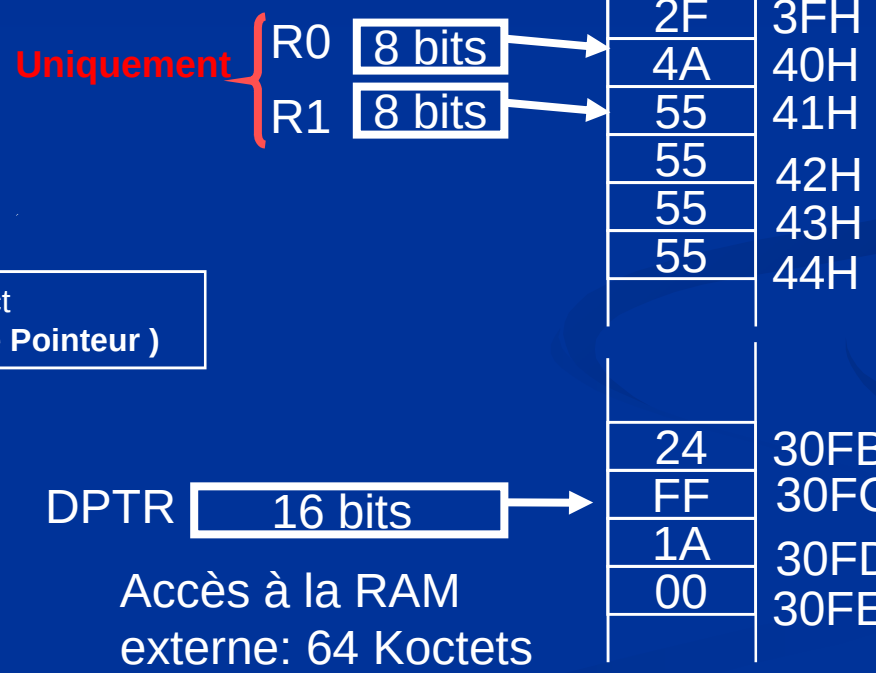
  MOV R3, A

- **Indirect par registre**

```
MOV     A,  # 0x56
MOV     0x40, A
MOV     R0, # 0x40
MOV     0x43, @R0
```

Symbole de l 'adressage indirect
Adresse Effective = (Registre Pointeur)

Registres Pointeurs



Accès à la RAM externe: 64 Koctets

Exemples:

```
BUFFER:     DS      100
MOV         R0,#BUFFER
MOV         A,@R0
INC         R0
MOV         @R0,A
```

```
XBUFFER:    DS    30FCh
VAR1:       DS      1
MOV         DPTR,#XBUFFER
MOVX        A,@DPTR
MOV         R1,#VAR1
MOVX        @R1,A
```

Indirect indexé par registre

MOVC A, @ A+PC
MOVC A, @ A+DPTR

Adresse Effective = (DPTR ou PC) + (A)
= (Base Adresse) + (Index)

Exemples:

```
TABLE:      DB      1,2,4,8,0x10
MOV         DPTR,#TABLE
MOV         A,#3
MOVC        A,@A+DPTR
```

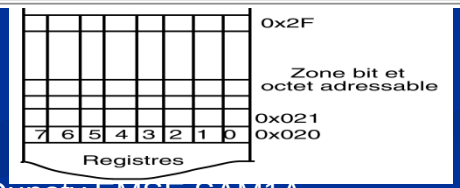
Bit adressage

Bit désigné par son poids de 0 à 7

UAL de bit

ANL C, ACC.7
MOV C, 0x21.3

0xF8	P5							
0xF0	B							
0xE8	P4							
0xE0	ACC							
0xD8	ADCON	ADDAT	DAPR					
0xD0	PSW							
0xC8	T2CON		CRCL	CRCH	TL2	TH2		
0xC0	IRCON	CCEN	CCL1	CCH1	CCL2	CCH2	CCL3	CCH3
0xB8	IEN1	IP1						
0xB0	P3							
0xA8	IEN0	IP0						
0xA0	P2							
0x98	SCON	SBUF						
0x90	P1							
0x88	TCON	TMOD	TL0	TL1	TH0	TH1		
0x80	P0	SP	DPL	DPH				PCON



Zone mémoire +
SFRs bits adressables

Exemples:

```
FLAG:       DBIT    1
P1          DATA    90H
GREEN_LED   BIT     P1.2

            SETB    GREEN_LED
            JB      FLAG,is_on
            SETB    FLAG
            CLR     ACC.5
             :
is_on:      CLR     FLAG
            CLR     GREEN_LED
```

Le registre d'état : Program Status Word (PSW)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CY	AC	F0	RS1	RS0	OV	UD	P

CY Carry flag, indique une retenue
AC Auxiliary Carry flag (pour les opérations BCD)
F0 Flag 0 (un indicateur à disposition de l'utilisateur)
RS1, RS0
Register bank select: RS1 RS0 Working Register Bank and Address

0 0 Bank0 (D:0x00 - D:0x07)
0 1 Bank1 (D:0x08 - D:0x0F)
1 0 Bank2 (D:0x10 - D:0x17)
1 1 Bank3 (D:0x18H - D:0x1F)

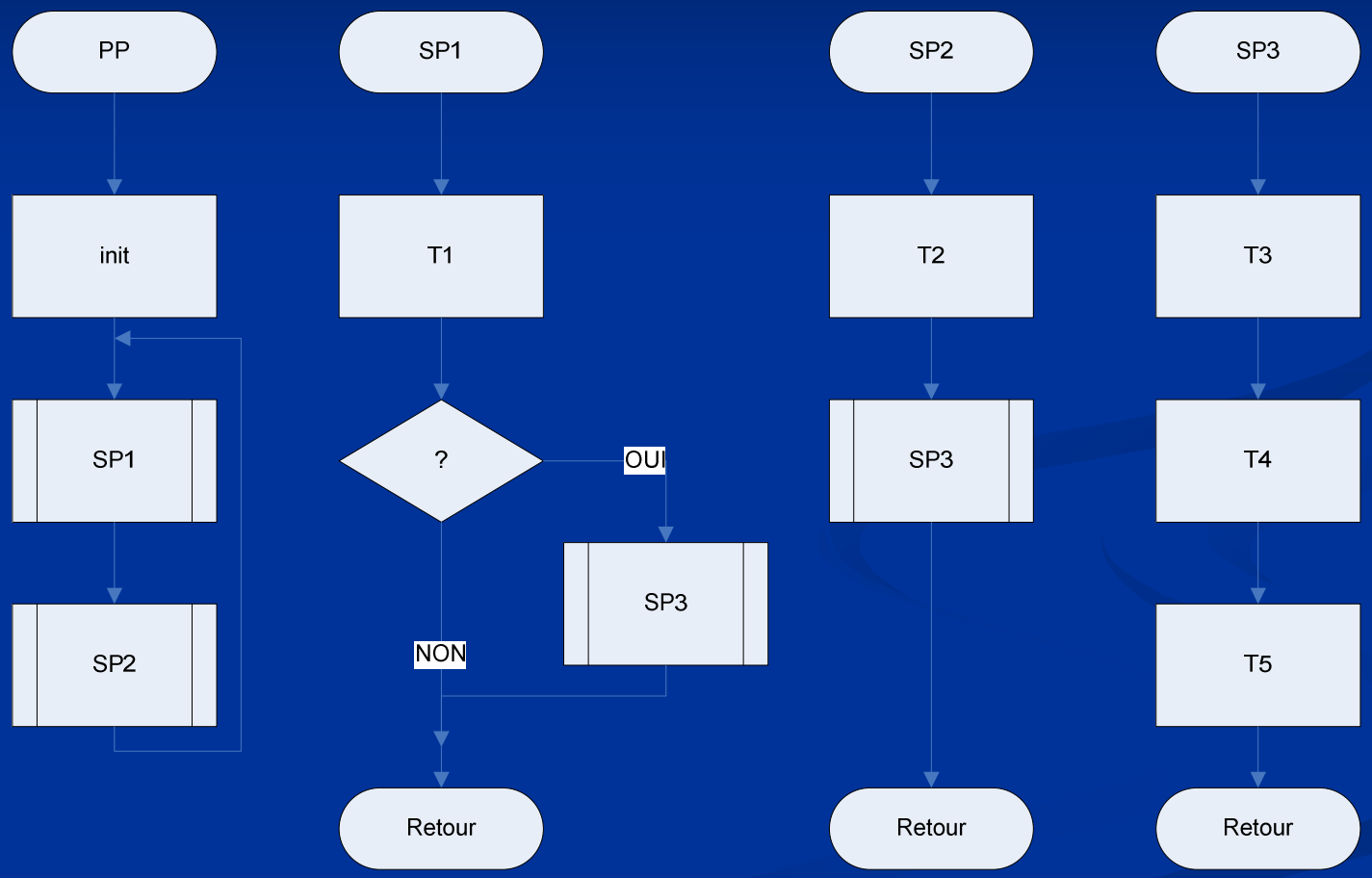
OV drapeau de débordement (Overflow flag)
UD User definable flag
P drapeau de parité (Parity flag)

```
jb  C,labas
jnb F0,encore
```

Les directives d'assemblage

Commentaire :	;		
Etiquette :		ici:	…
Equivalence :	EQU	var1	EQU 123
Réservation en RAM :	DS	label :	DS 5
Initialisation de constantes :	DB	ici :	DB 27,33h,'coucou'
Initialisation de const 16bits :	DW	la :	DW 123 ,ABCDh
Réservation d'un bit	DBIT	flag :	DBIT 2
Adresse courante	$		jmp $
Fin du texte assembleur			END

Programmation structurée



Sous-programmes, la pile S

L'instruction <u>call</u> est codée sur deux octets
call tempo se trouve à l'adresse 0x123
call calcul se trouve à l'adresse 0x456

```
mov     s,#7Fh
call    tempo
…
jmp     ailleurs

tempo: …
call    calcul
…
ret

calcul:…
…
ret
```

```
call     calcul

call     tempo

mov      s,#7Fh
```

AD RAM pointée par SP	Valeur
0x84	?
0x83	58h
0x82	04h
0x81	25h
0x80	01h
0x7F	?

ret

ret

La définition des segments mémoires

- DATA : mémoire RAM et SFR de 0x00 à 0xFF, adressable en direct et indirect
- BIT : zone RAM « bit adressable » et certains SFR, adressables avec les instructions pour bits.
- IDATA : RAM adressable en indirect par les registres R0,R1, (généralement 0x00 à 0xFF)
- XDATA : RAM externe accessible par l'instruction MOVX via le registre DPTR.
- CODE : ROM accessible via MOVC et le registre DPTR

- Nommer un segment : SEGMENT
- Sélectionner un segment : RSEG
- Segment pré-definis : CSEG (ROM), DSEG (DATA), BSEG(BIT), ISEG (IDATA, XSEG (XDATA)

- La directive ORG permet de changer d'adresse DANS un segment

Structure mémoires

Utilisation du Linker La définition des segments

```
mesdonnees      SEGMENT DATA   ; segment de DATA
monprog         SEGMENT CODE   ; ROM
mesconst        SEGMENT CODE   ; ROM
pile            SEGMENT IDATA  ; place pour la pile en IDATA

       RSEG    mesdonnees
val :  DS       1            ; réserve un octet en RAM DATA

       RSEG    pile         ; reservation pour pile en IDATA
       DS      20h

       CSEG    AT 0          ; autre methode
       LJMP    debut         ; saut sur debut
               …
```

Suite

```
        RSEG    monprog       ; zone ROM connue du linker
debut:  MOV     SP,#pile-1
        mov     a,#0ABh
        mov     val,a
…
        RSEG    mesconst      ; une autre zone ROM
Const:  DB      27,33h,'coucou',0

        RSEG    mesdonnees    ; on se place en RAM
Donnee :      DS      20          ; reserve 20 octets en RAM

        RSEG    monprog     ; suite du programme en ROM
        inc     a
        mov     R0,val
        mov     @R0,a
        jmp     $
```

QUIZZ

- A quoi sert l'accumulateur, le Program Counter ?
- Le PC pointe toujours sur une instruction ?
- Dans quel registre se trouve les Flags liés à l 'A.L.U ?
- Pour obtenir une capacité d 'adressage de 128 Koctets combien de lignes d 'adresse faut-il ?
- Une Pile LIFO peut se placer n 'importe ou dans l 'espace mémoire du microcontrôleur.
- C 'est au programmeur de gérer les déplacements pointeur de pile.
- En quoi un débordement de pile est catastrophique.
- Le bus d 'adresses permet d 'accéder à la fois aux constantes et au code, mais pas aux périphériques.
- Quels sont les avantages de la programmation structurée
- Un registre est une case mémoire accessible comme une autre
- Pourquoi dans les systèmes embarqués on utilise des RAMs, des EPROMs et pas des disques durs.
- Intérêt des mémoires FLASH ?, remplacent-t-elle les EEPROMs ?
- Dans un prcesseur CISC un cycle d 'instruction machine correspondant-il toujours à une période d 'horloge ?

- Différence entre MOV, MOVC ?

- Quel est l 'état des registres R0, R1, A, B, du port 1 et de la case mémoire 40H , après l 'exécution du programme suivant ?

```
MOV     R0, #30H
MOV     A,  @R0
MOV     R1, A
MOV     B, @R1
MOV     @R1, P1
MOV     P2, P1
```

Donnees	Adresses
0CAH	P1
40 H	30H
...	
10H	40H

- Quel est l 'état de la carry, du port 1 et de la case mémoire 30H après l 'exécution du programme suivant ?

```
SETB    C
MOV     P1.3, C
MOV     C, P3.3
MOV     P1.2, C
CLR     C
MOV     30H.6, C
```

11000101B	P3
35H	P1
40 H	30H
...	

- Il y a une erreur dans le programme précédent, laquelle ?

- Quelles sont les valeurs des registres DPH et DPL après l 'instruction suivante ?

```
MOV     DPTR, #4660
```

- Quel est l 'état des cases mémoires ci-dessous après l 'exécution du programme suivant ?

Donnees	Adresses
40 H	1FFFH
...	
0CDH	2FA0H

Mémoire Programme

Donnees	Adresses
1F H	1FFFH
...	
00H	2FA0H

Mémoire Donnée

Initialisation routine

```
MOV DPSEL, #06H
MOV DPTR, #1FFFH
MOV DPSEL, #07H
MOV DPTR, #2FA0H
```

Table transfert routine

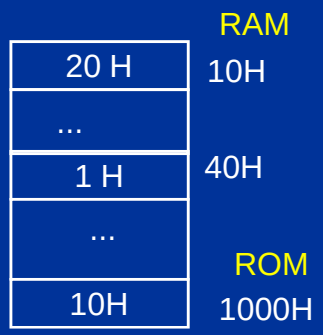
```
CLR  A
MOV DPSEL, #06H
MOVC     A, @A + DPTR
MOV DPSEL, #07H
MOVX     @DPTR, A
```

- Corriger les erreurs

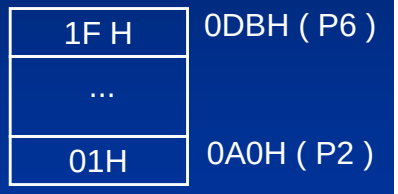
```
MOV     DPTR,#0x1FFF0
MOVC    A,@DPTR
MOV     DPTR, #0x10  ;pointeur en RAM interne
MOV     A,@A+DPTR
MOV     R0,#0x1000
MOV     DPTR,#FA20H
```

- Quelle est la valeur contenu dans l 'accumulateur à la fin du programme suivant:

```
MOV     DPTR,#1000H
MOVC    A,@A+DPTR
MOV     R0, A
MOV     A, #2
ADD     A, R0
ADD     A, #2
ADD     A,@R0
ADD     A, 40H
```



- Quelle est la valeur contenu dans l 'accumulateur à la fin du programme suivant ?




```
MOV     R1, #0DBH
MOV     A, 0A0H
ADD     A,@R1
```

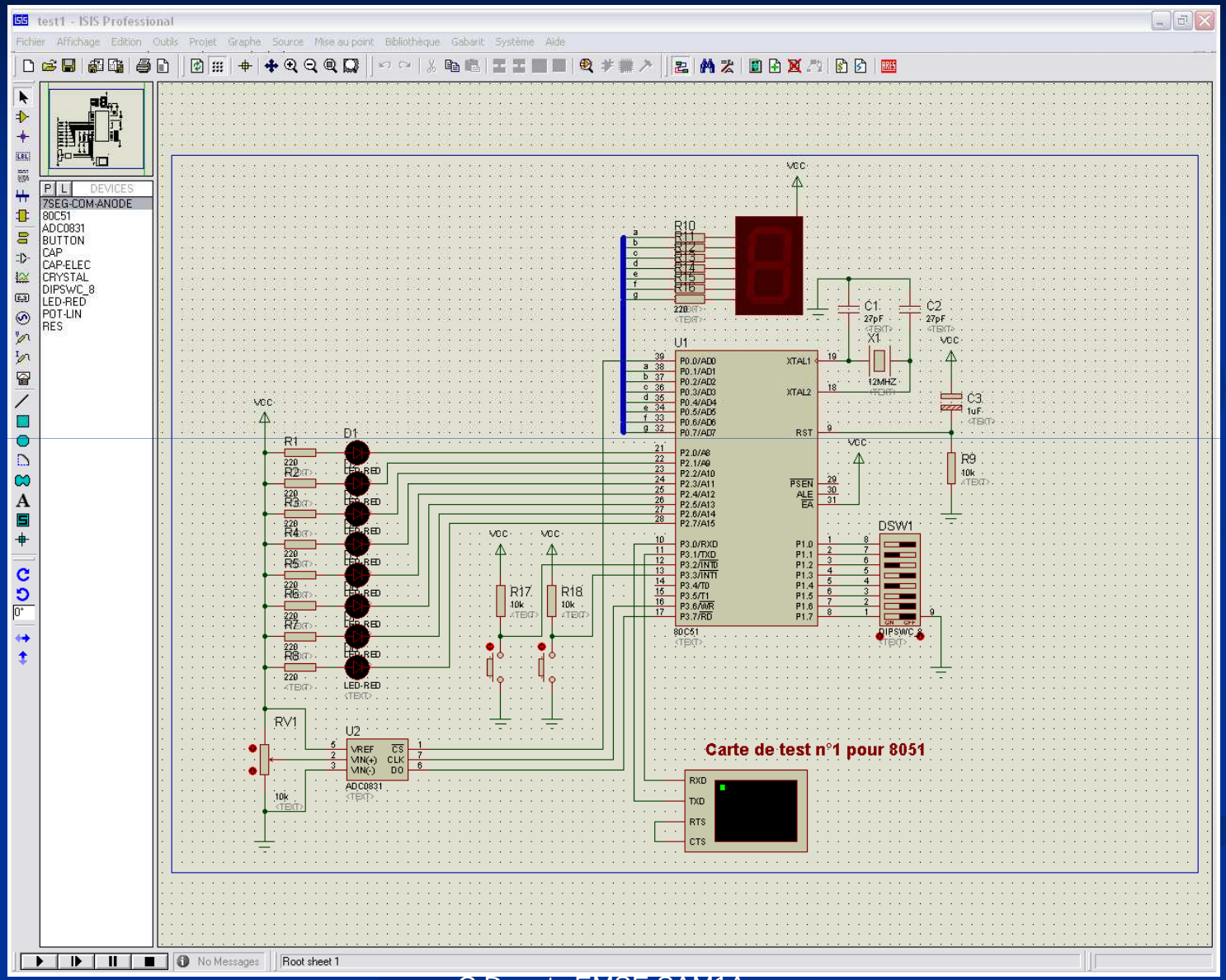
- Qu 'appelle t-on vecteur de reset ?

- Quelle est la directive assembleur qui permet de placer le code sur le vecteur de RESET?

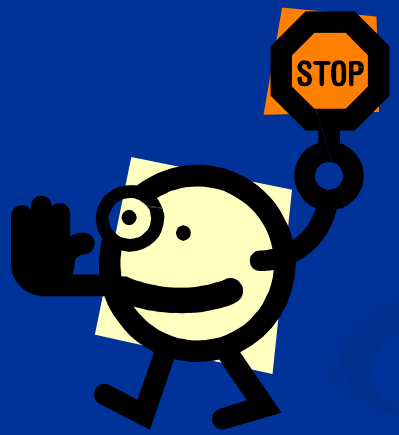
TP N°2

Utiliser les outils de développement
Découvrir le jeu d'instruction
Créer et tester des programmes simples

ISIS VSN & uVISION

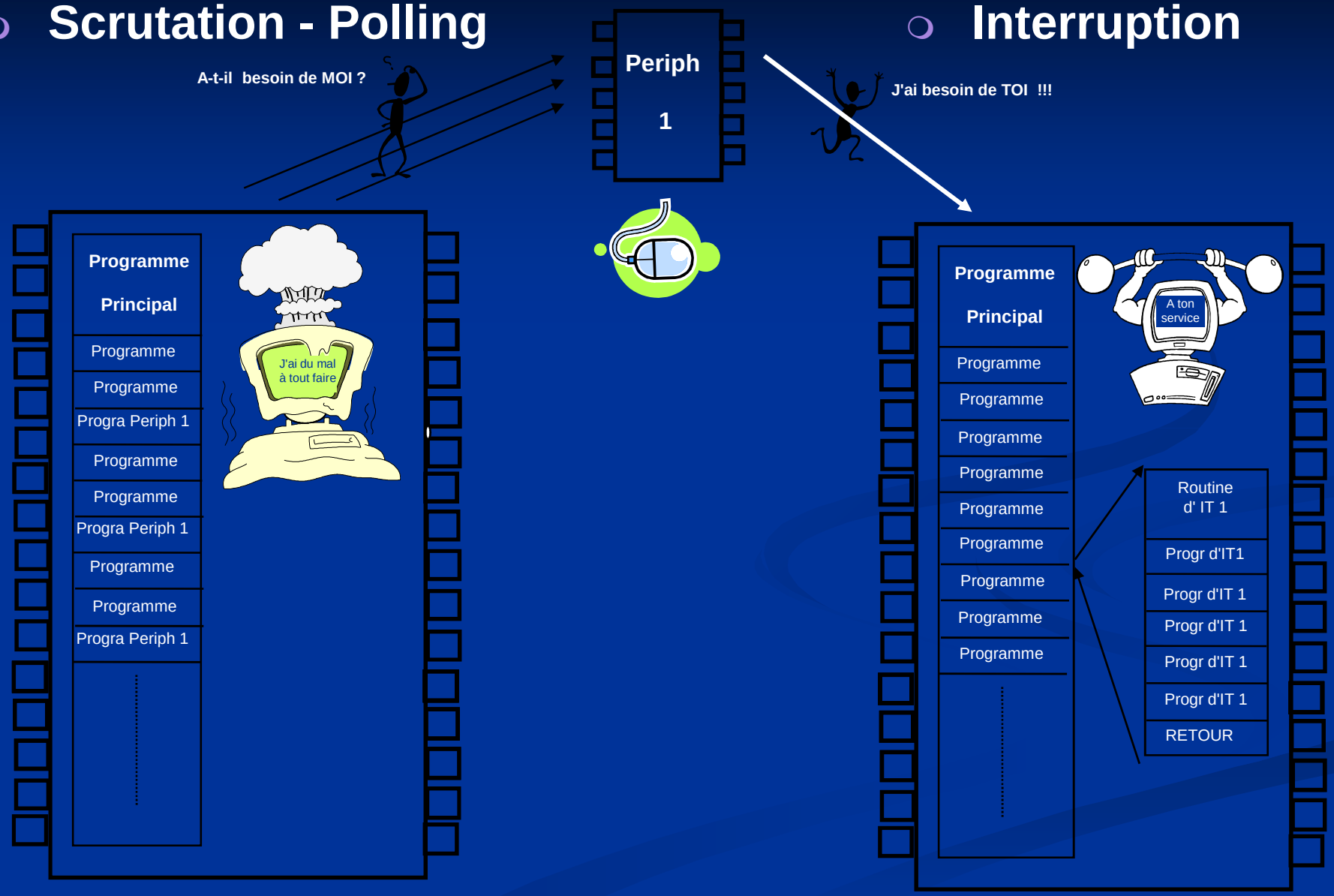


Les interruptions

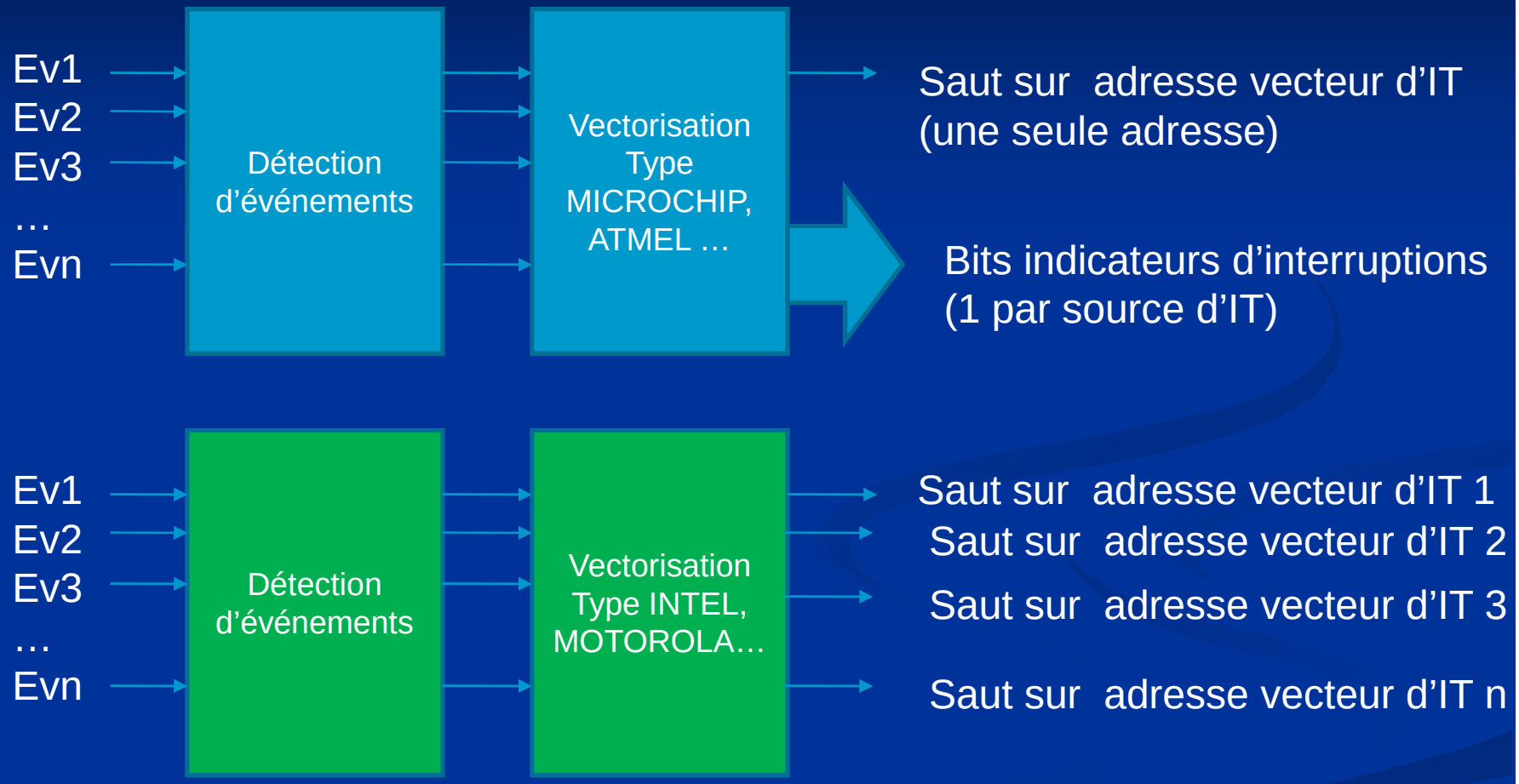


Scrutation VS Interruptions

- **Scrutation - Polling**
- **Interruption**



Deux approches



interruptions

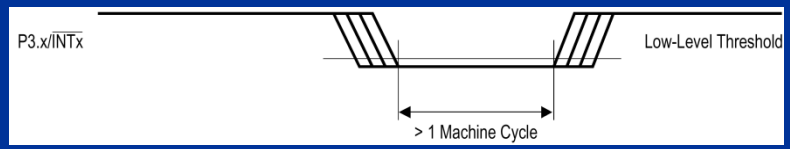
- **Validation globale**

  Inhibition ou Validation de toutes les interruptions

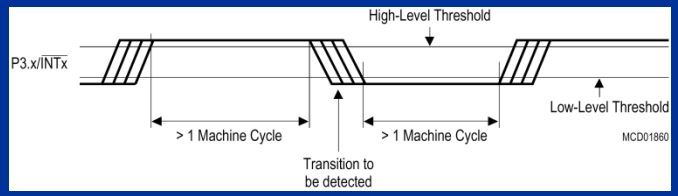
- **Niveau de priorité**

  Interruptible si priorité N+1

- **Type de déclenchement**
  - **Sur niveau**



  - **Sur front**



- **Routine d 'interruption**

  Sous-programme d 'IT

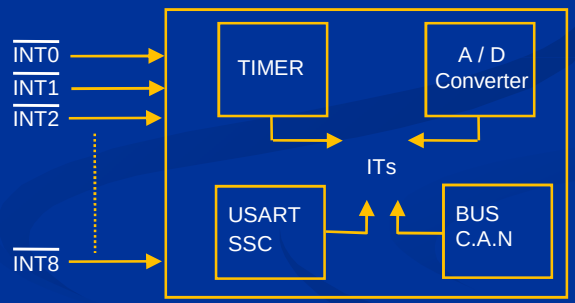
- **Masquable**

  Une interruption masquable peut être mise hors service par le microcontroleur

- **Non Masquable**

  Une interruption non masquable (NMI) ne peut être ignorée

- **Externe, Interne**

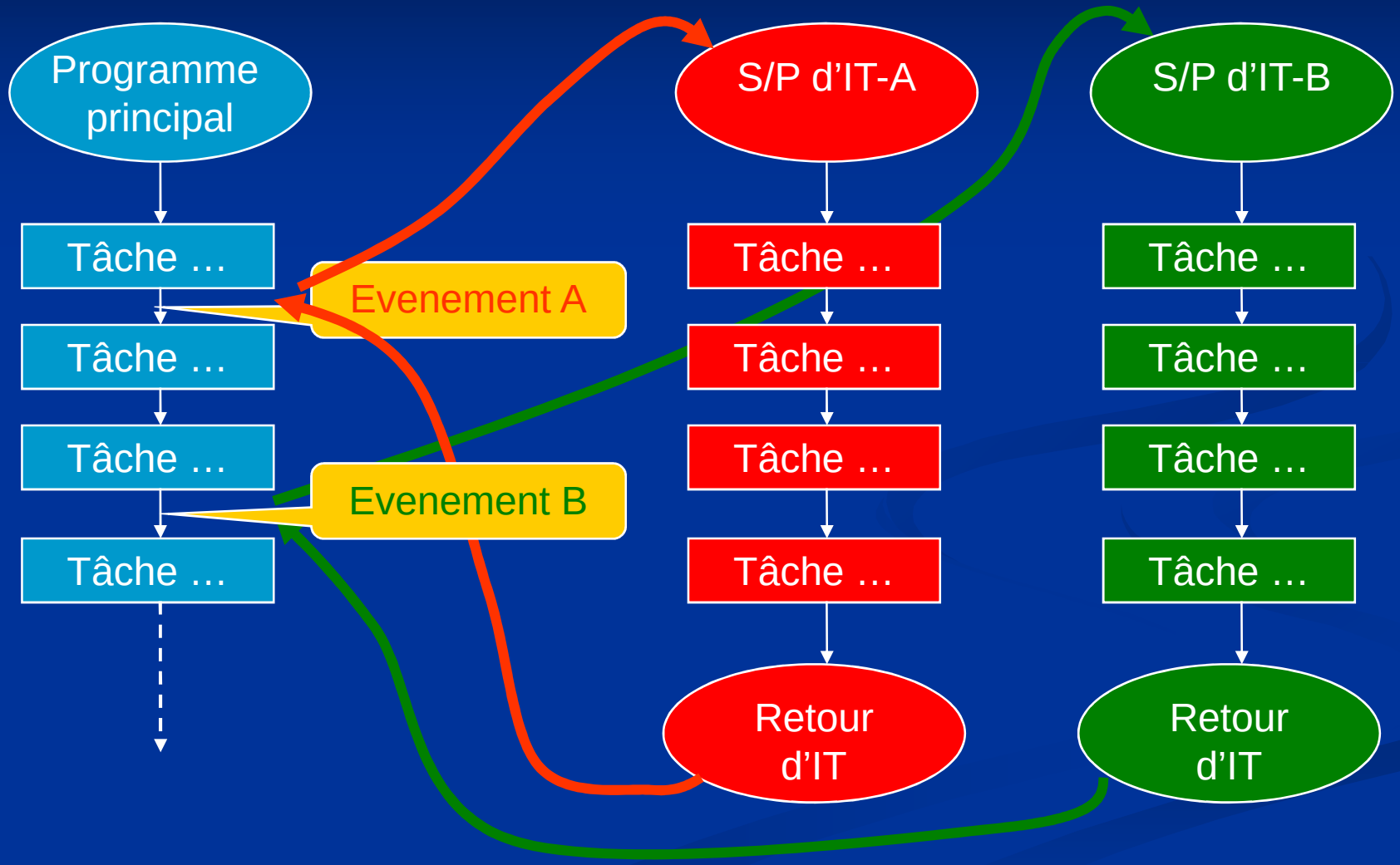


- **Acquittement**
  - **Matériel**
  - **Logiciel**

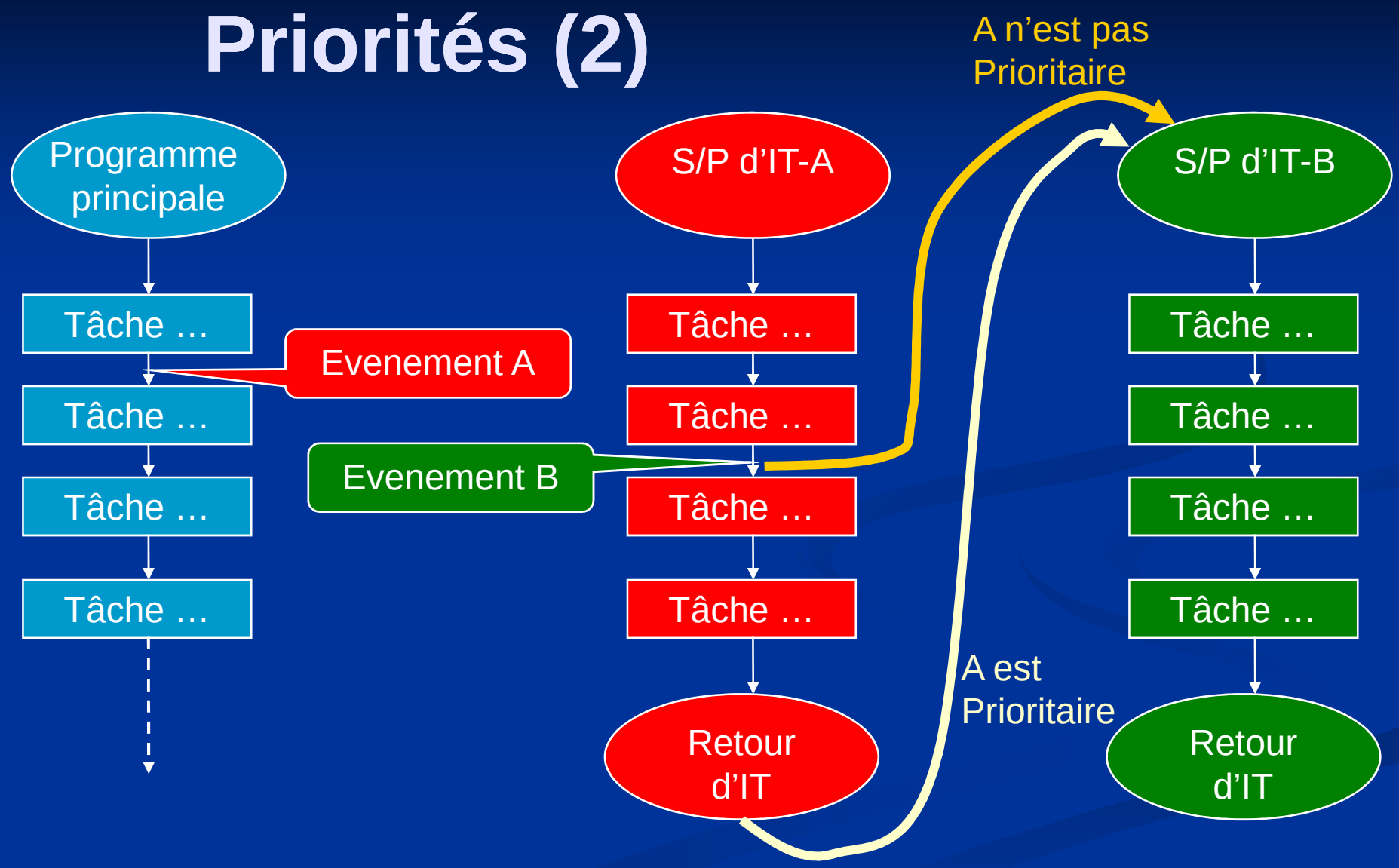
- **Vecteur**

  N° Vecteur ----> Vecteur

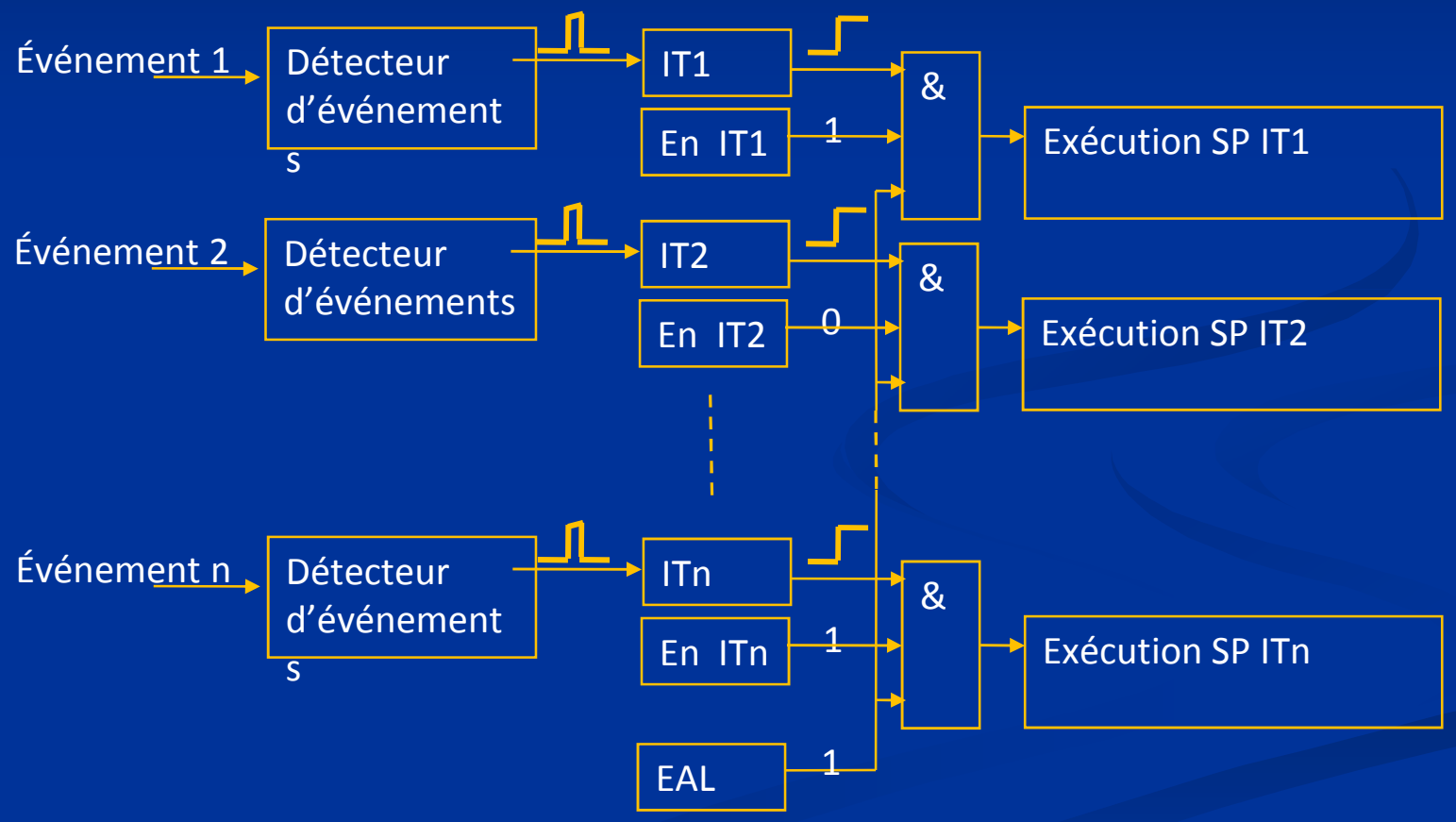
Interruptions – Priorités (1)



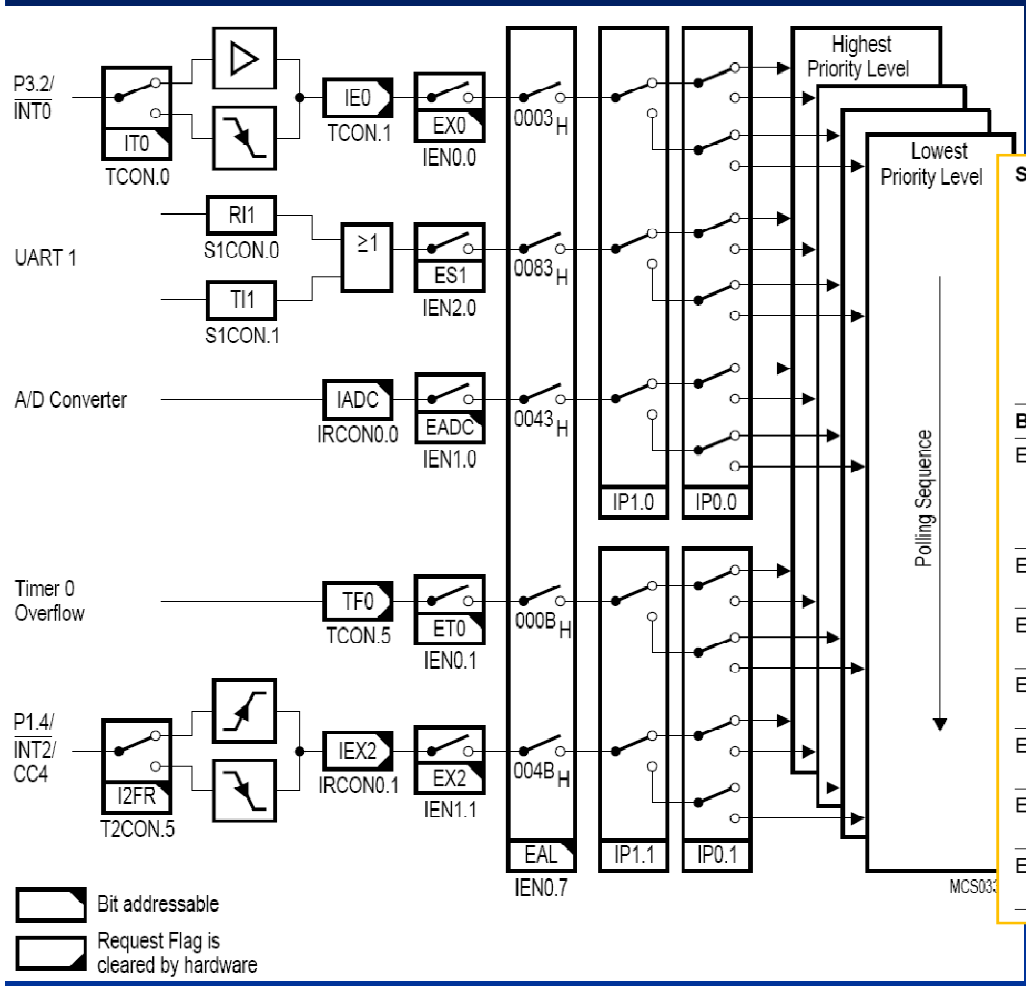
Interruptions – Priorités (2)



Masquage / Validation



IT sur C517A



Special Function Register IEN0 (Address $A8_H$) **Reset Value : 00_H**

Bit No.	MSB AF_H	AE_H	AD_H	AC_H	AB_H	AA_H	$A9_H$	LSB $A8_H$	
$A8_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0

The shaded bit is not used for interrupt control

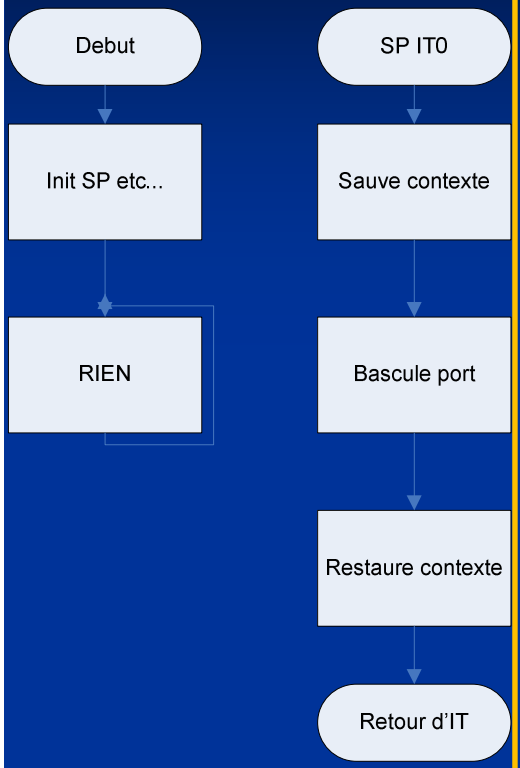
Bit	Function
EAL	Enable/disable all interrupts. If EA=0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
ET2	Timer 2 interrupt enable. If ET2 = 0, the timer 2 interrupt is disabled.
ES0	Serial channel 0 interrupt enable If ES0 = 0, the serial channel interrupt 0 is disabled.
ET1	Timer 1 overflow interrupt enable. If ET1 = 0, the timer 1 interrupt is disabled.
EX1	External interrupt 1 enable. If EX1 = 0, the external interrupt 1 is disabled.
ET0	Timer 0 overflow interrupt enable. If ET0 = 0, the timer 0 interrupt is disabled.
EX0	External interrupt 0 enable. If EX0 = 0, the external interrupt 0 is disabled.

Sources et vecteurs d'IT du C517A

8051 générique

Interrupt Source	Interrupt Vector Address	Interrupt Request Flags
External Interrupt 0	0003H	IE0
Timer 0 Overflow	000BH	TF0
External Interrupt 1	0013H	IE1
Timer 1 Overflow	001BH	TF1
Serial Channel 0	0023H	RI0 / TI0
Timer 2 Overflow / Ext. Reload	002BH	TF2 / EXF2
A/D Converter	0043H	IADC
External Interrupt 2	004BH	IEX2
External Interrupt 3	0053H	IEX3
External Interrupt 4	005BH	IEX4
External Interrupt 5	0063H	IEX5
External Interrupt 6	006BH	IEX6
Serial Channel 1	0083H	RI1 / TI1
Compare Match Interupt of Compare Registers CM0-CM7 assigned to Timer 2	0093H	ICMP0 - ICMP7
Compare Timer Overflow	009BH	CTF
Compare Match Interupt of Compare Register COMSET	00A3H	ICS
Compare Match Interupt of Compare Register COMCLR	00ABH	ICR

Programme d'IT : demoIT.a51




```
CSEG    AT      0
        ljmp    prog

        ORG     3       ; vecteur INT0
        push    acc     ; pour demo
        push    psw
        cpl     P2.0    ; bascule P2.0
        clr     IE0     ; efface drapeau
        pop     psw
        pop     acc
        reti

prog:   mov     sp,#7Fh
        setb    IT0   ; detection front
desc
        setb    EX0   ; active IT0
        setb    EAL  ;active les IT
        sjmp    $
        END
```

TP N°3

Essayer un programme fonctionnant en interruption
Créer un programme gérant plusieurs sources d'interruptions

TIMER / COMPTEUR

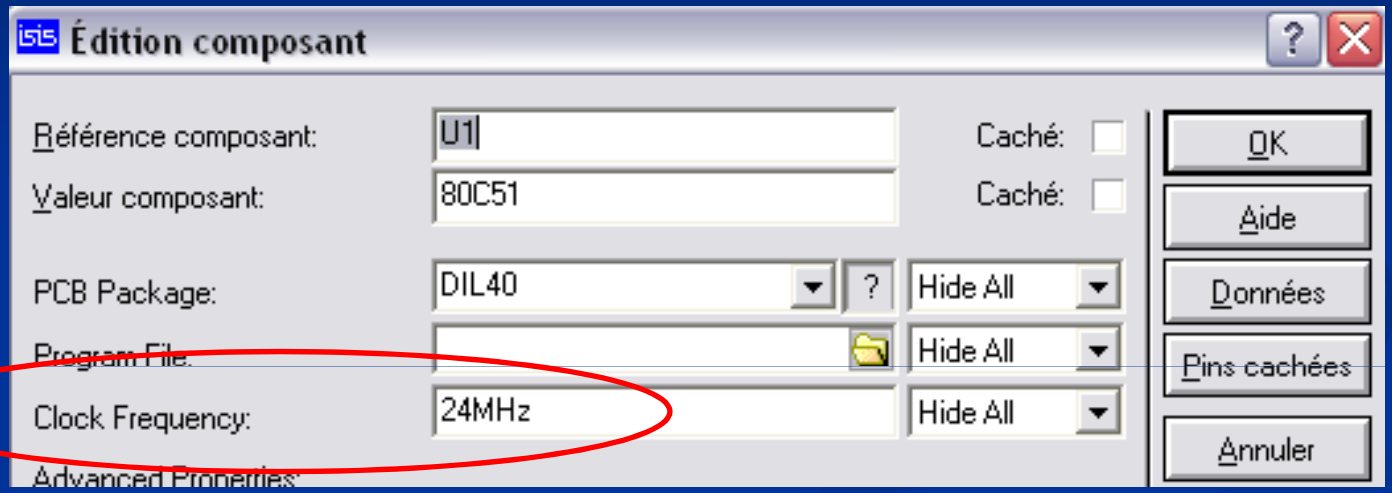
Les microcontrôleurs intègrent des compteurs (8 ou 16 bits) qui suivant leur utilisation sont nommé TIMER ou COMPTEUR.

Utilisation comme TIMER : généralement lorsque leur horloge est « constante » ils peuvent alors mesurer ou produire des « temps ».

Utilisation comme COMPTEUR : Généralement ils comptent des événements (fronts montants ou descendant) sur une entrée physique du microcontrôleur.

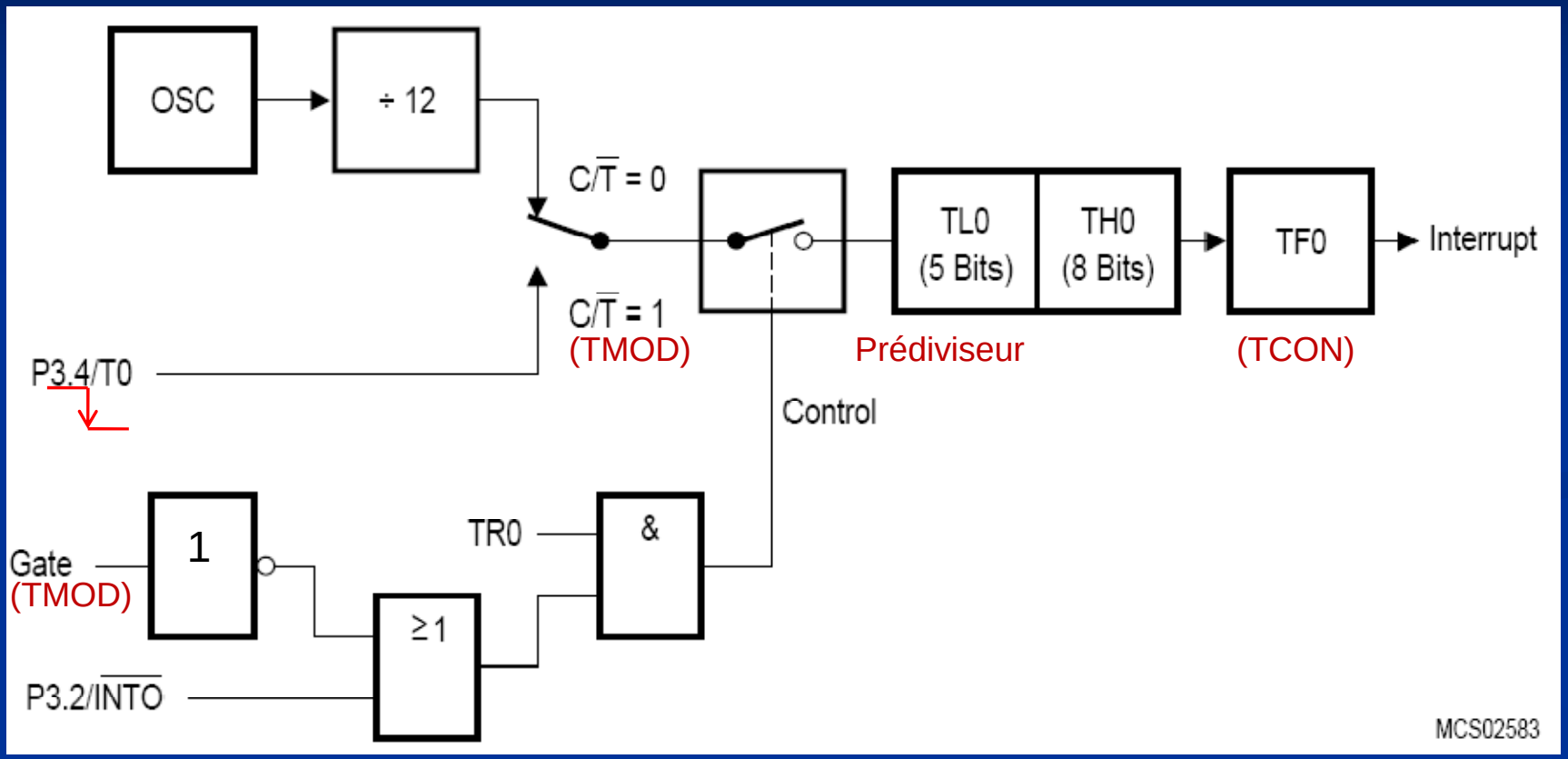
Le 8051 possède trois TIMER/COUNTER avec des applications différentes pour chaque

Choisir la fréquence de l'oscillateur

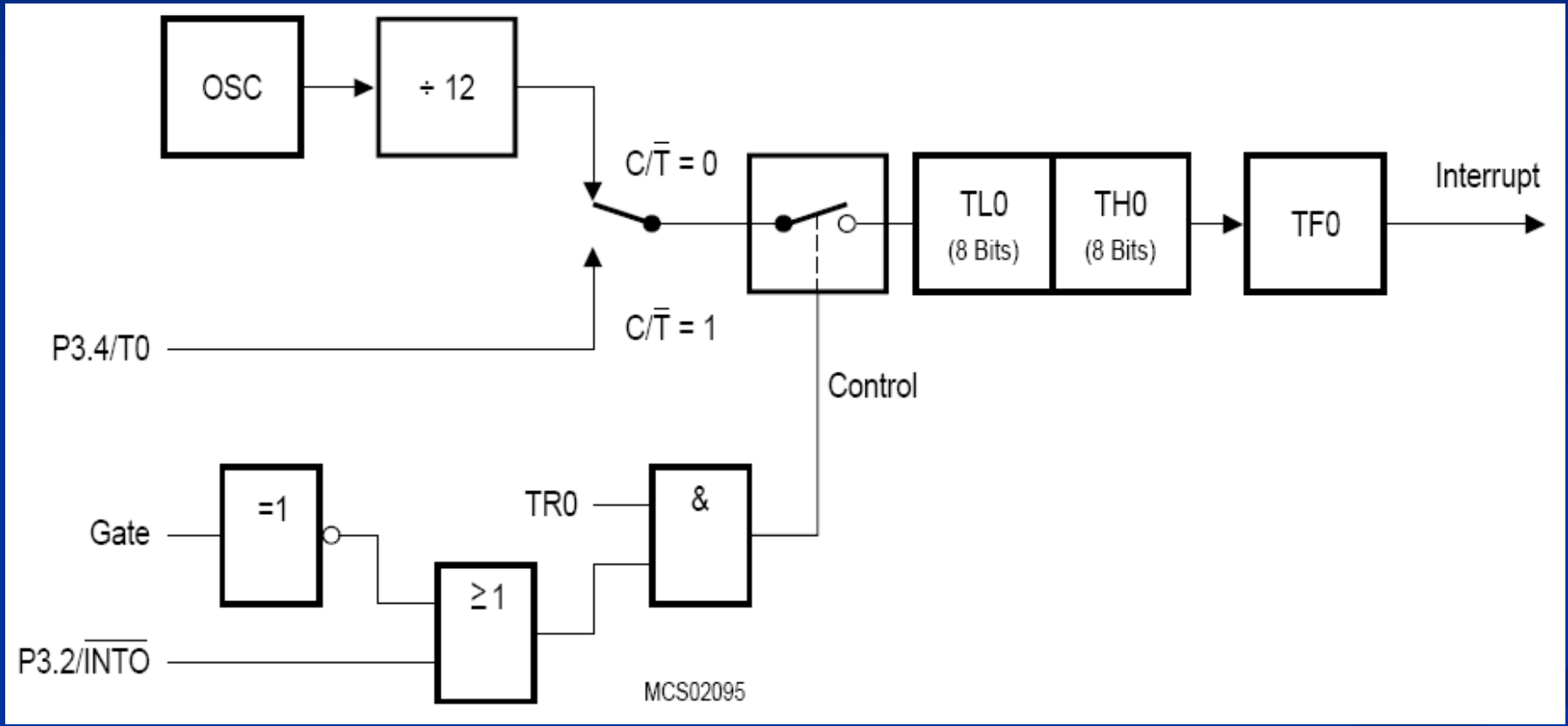


Lorsque ISIS-VSM est utilisé comme debugger, la fréquence de l'oscillateur n'est plus définie dans uVISION mais dans les propriétés ISIS du 8051 (clic-droit puis EDITER PROPRIETES)

MODE 0 pour TIMER 0 et 1



MODE1 pour TIMER 0 et 1



Le registre TCON

Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
	$8F_H$	$8E_H$	$8D_H$	$8C_H$	$8B_H$	$8A_H$	89_H	88_H	
88_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON

The shaded bits are not used in controlling timer/counter 0 and 1.

Bit	Function
TR0	Timer 0 run control bit Set/cleared by software to turn timer/counter 0 ON/OFF.
TF0	Timer 0 overflow flag Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	Timer 1 run control bit Set/cleared by software to turn timer/counter 1 ON/OFF.
TF1	Timer 1 overflow flag Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.

Le registre TMOD

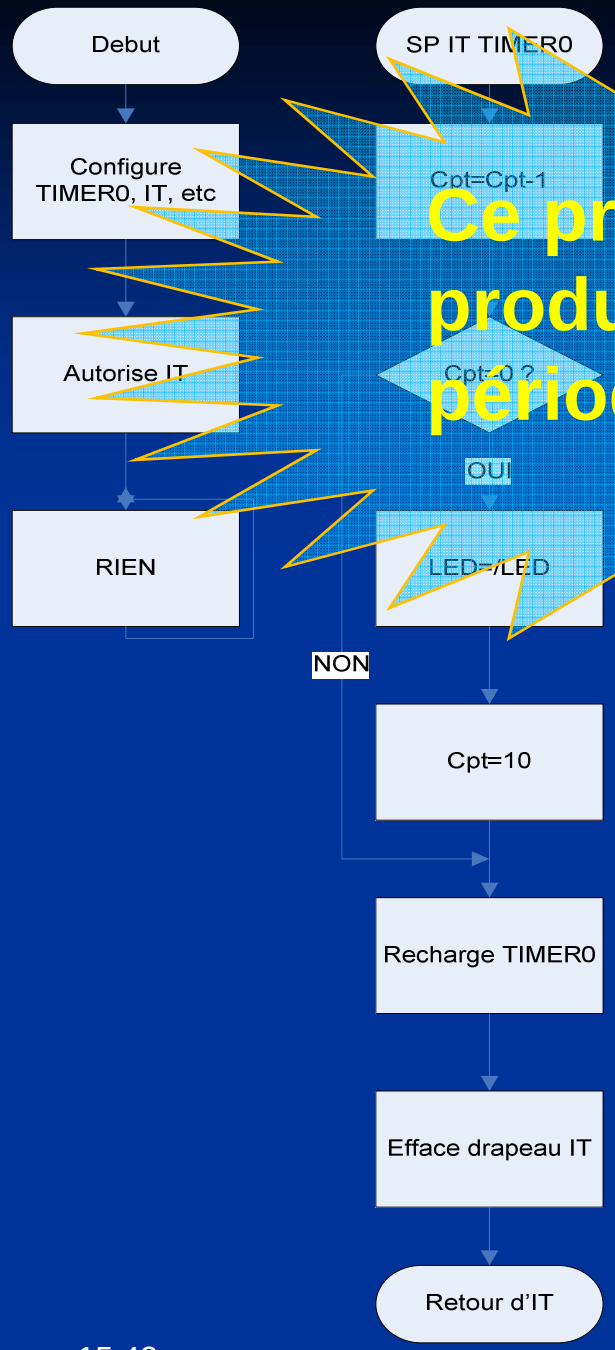
Special Function Register TMOD (Address 89_H) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
89_H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	TMOD
	Timer 1 Control				Timer 0 Control				

Bit	Function
GATE	Gating control When set, timer/counter "x" is enabled only while "INT x" pin is high and "TRx" control bit is set. When cleared timer "x" is enabled whenever "TRx" control bit is set.
C/$\overline{T}$	Counter or timer select bit Set for counter operation (input from "Tx" input pin). Cleared for timer operation (input from internal system clock).
M1 M0	Mode select bits

M1	M0	Function
0	0	8-bit timer/counter: "THx" operates as 8-bit timer/counter "TLx" serves as 5-bit prescaler
0	1	16-bit timer/counter. "THx" and "TLx" are cascaded; there is no prescaler
1	0	8-bit auto-reload timer/counter. "THx" holds a value which is to be reloaded into "TLx" each time it overflows
1	1	Timer 0 : TL0 is an 8-bit timer/counter controlled by the standard timer 0 control bits. TH0 is an 8-bit timer only controlled by timer 1 control bits. Timer 1 : Timer/counter 1 stops

Exemple mode 1: flash_it.a51

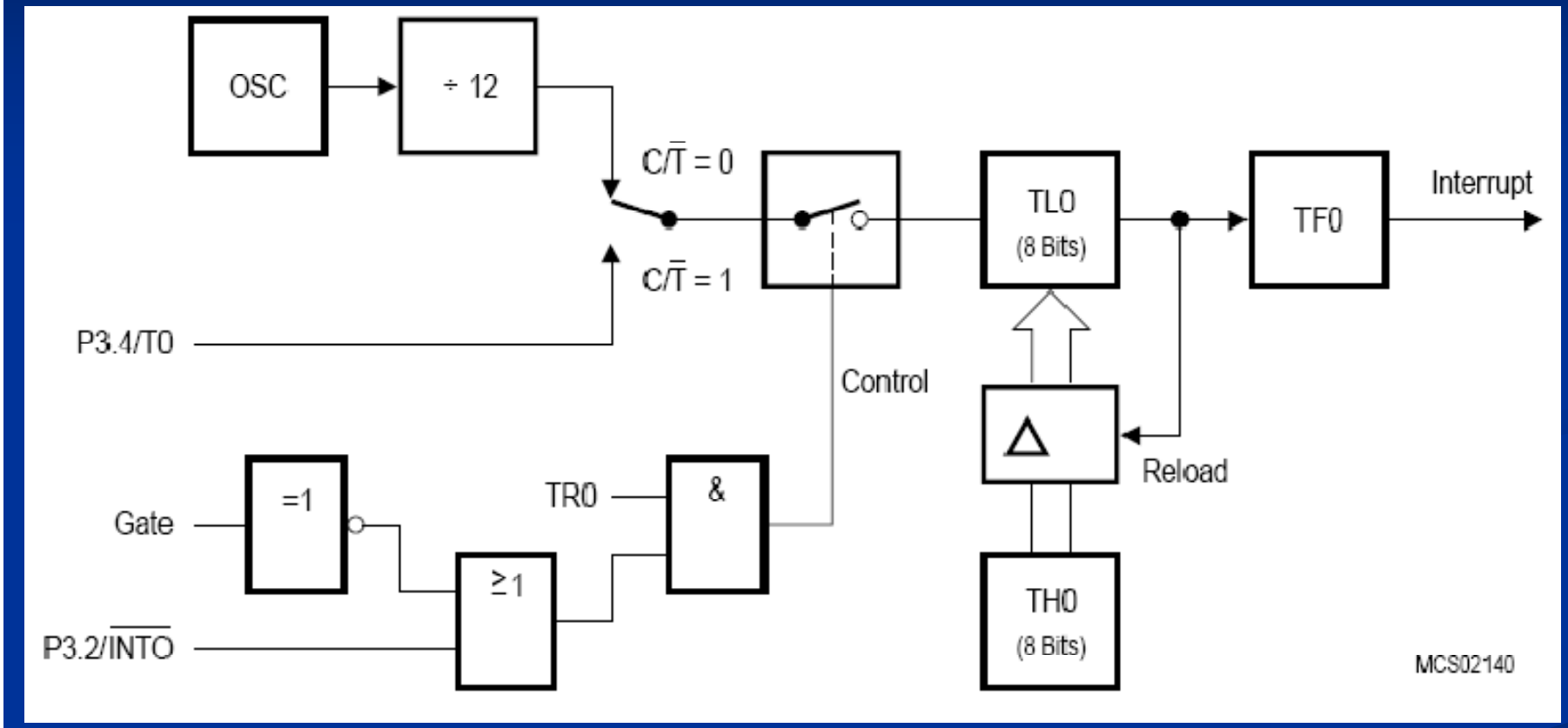


Ce programme ne produira pas une période de 500ms !

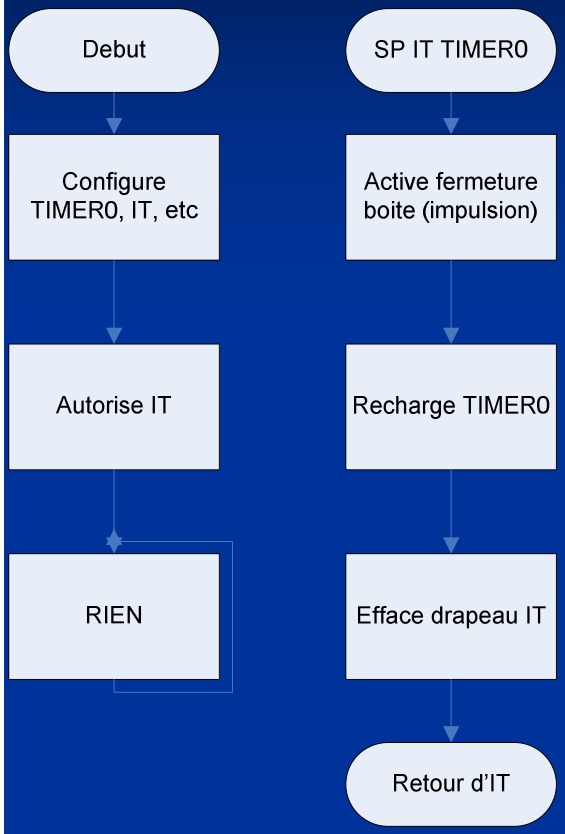
```
valTH0  EQU 3Ch ; 3CB0 = FFFFh-50000d+1
valTL0  EQU 0B0h
CPT     EQU 20h
        CSEG    AT 0
        LJMP    PROG
        ORG     0x000B  ; vecteur TIMER0 ovf
        LJMP    IT_TIMER0
PROG:   MOV     SP,#STACK-1
        MOV     TMOD,#00000001b
        SETB    TR0     ; run TIMER 0
        MOV     CPT,#10     ; compteur d'IT
        SETB    ET0     ; IT sur OVF TIMER0
        SETB    EAL     ;
        SJMP    $       ; attend ;
IT_TIMER0:
        DJNZ    CPT,NON
        CPL     LED     ; bascule LED
        MOV     CPT,#10 ;recharge cpt d'IT
NON:    MOV     TH0,#3Ch    ; recharge TIMER0
        MOV     TL0,#0AFh
        CLR     TF0     ; efface drapeau IT0
        RETI
```

FLASH sur P2.0, BdT TIMER0 mode 1 , periode 0,5s (demi periode 250 mS)
Bdt=500nS : comptage : 50000 pulses soit 25mS,
FFFFh-50000d= 15535d = 3CAFh
Bascule toutes les 10 interruptions soit 250mS

MODE 2 pour TIMER 0 et 1



Exemple 2 Comptage d'événements demo_cpt_MODE1.A51

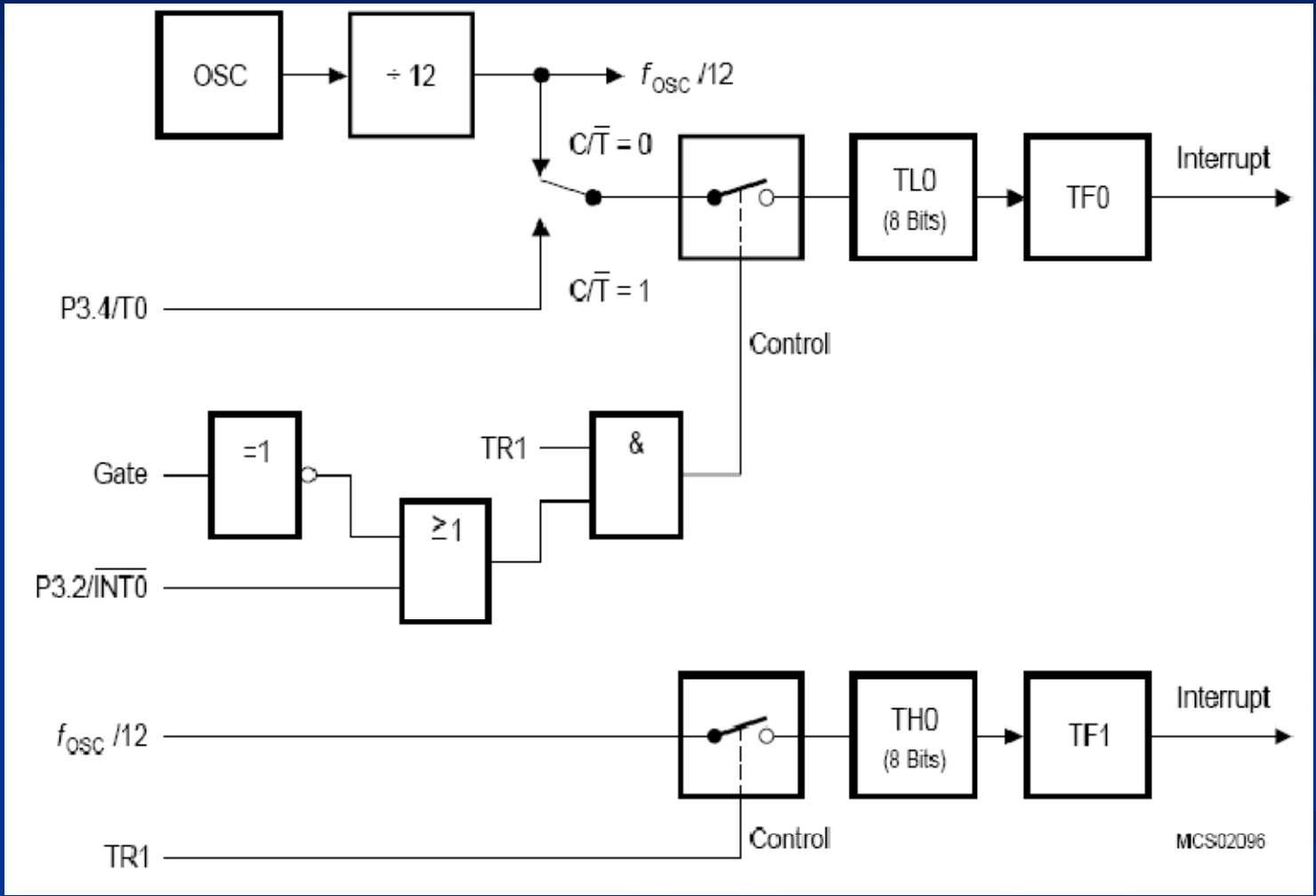



```
; 10 bonbons par boite
valTH0 EQU        0FFh
valTL0 EQU        0F6h
BOITE  EQU        P2.0 ; fermeture boite
STACK  SEGMENT    IDATA
       RSEG       STACK
       DS         32
CPT:   DS         1      ; compteur
       CSEG     AT 0
       LJMP     start
       ORG 0x000B     ; vecteur T0 ovf
       LJMP   IT_TIMER0
…
```

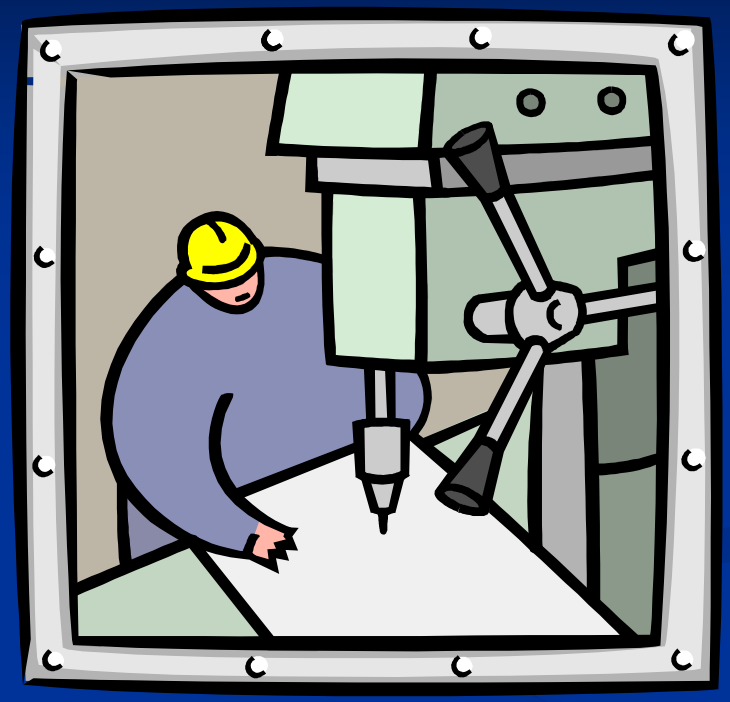
Un capteur optique détecte le passage de bonbons sur un tapis roulant les entrainant dans une boite. La boite doit être fermée, lorsqu'elle contient dix bonbons.

```
start:        MOV    SP,#STACK-1
              MOV    TMOD,#00000101b ; TIMER 0 sur mode 1
source Q/12
              MOV    CPT,#0 ; compteur de boites =0
              MOV    TH0,#valTH0  ; recharge TIMER0
              MOV    TL0,#valTL0
              SETB   ET0    ; autorise IT sur TIMER0
              SETB   EAL    ; autorise toutes les IT
              SETB   TR0    ; run TIMER 0
              SJMP   $      ; attend IT
;
IT_TIMER0:    SETB   BOITE  ; impulsion sur fermeture
              CLR    BOITE
              INC    CPT    ;incrémente compteur de boites
              MOV    TH0,#valTH0  ; recharge TIMER0
              MOV    TL0,#valTL0
              CLR    TF0    ; efface drapeau IT0
              RETI
             END              ; End Of File
```

MODE 3

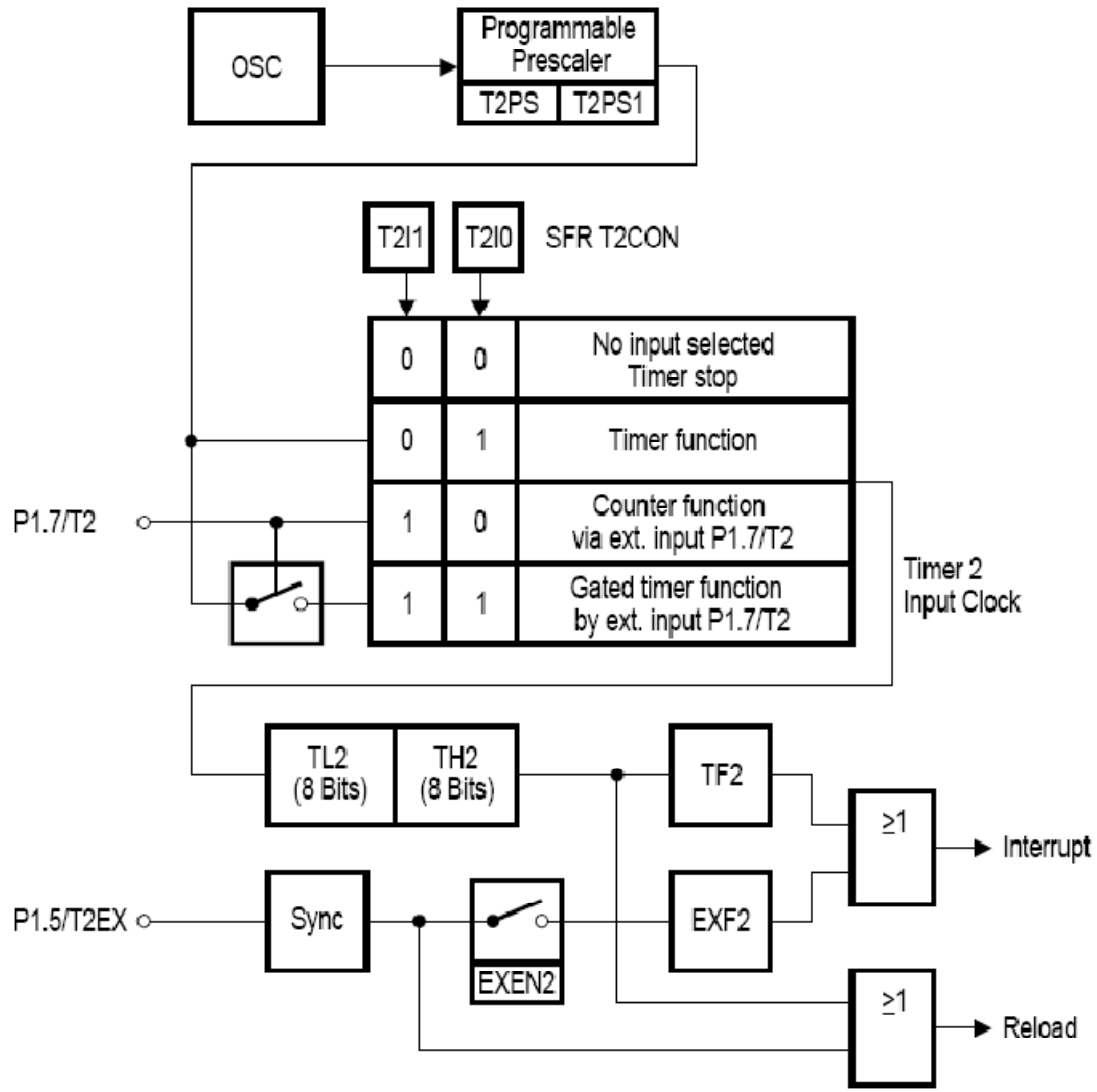

OSC
÷ 12
f_{osc} /12
C/T̄ = 0
TL0
(8 Bits)
TF0
Interrupt
P3.4/T0
C/T̄ = 1
Control
Gate
=1
TR1
&
≥1
P3.2/INT0
f_{osc} /12
TH0
(8 Bits)
TF1
Interrupt
TR1
Control
MCS02096

TP N°4



Mise en œuvre des TIMERS
comptage
production de signaux

TIMER 2



Les fonctions matérielles CAPTURE/COMPARE permettent de limiter considérablement le logiciel. **COMPARE** automatisent les actions sur les broches externes. (disponible sur certains 8051)
CAPTURE facilite l'échantillonnage temporel pour la mesure de durée.

Les fonctions CAPTURE/COMPARE s'appuient sur le TIMER 2 ou le « COMPARE TIMER »

T2I1, T2I0 =0,1 le TIMER 2 compte l'horloge issue du prédiviseur OSC/12 /24 /48 /96
T2I1, T2I0 =1,0 le COUNTER 2 compte les fronts descendants sur T2
T2I1, 0T2I0 =1,1 le TIMER 2 compte l'horloge issue du prédiviseur OSC/12 (uniquement) lorsque T2=1
En cas de débordement TF2=1 une interruption peut être activée.

TIMER2
Le registre TCON2

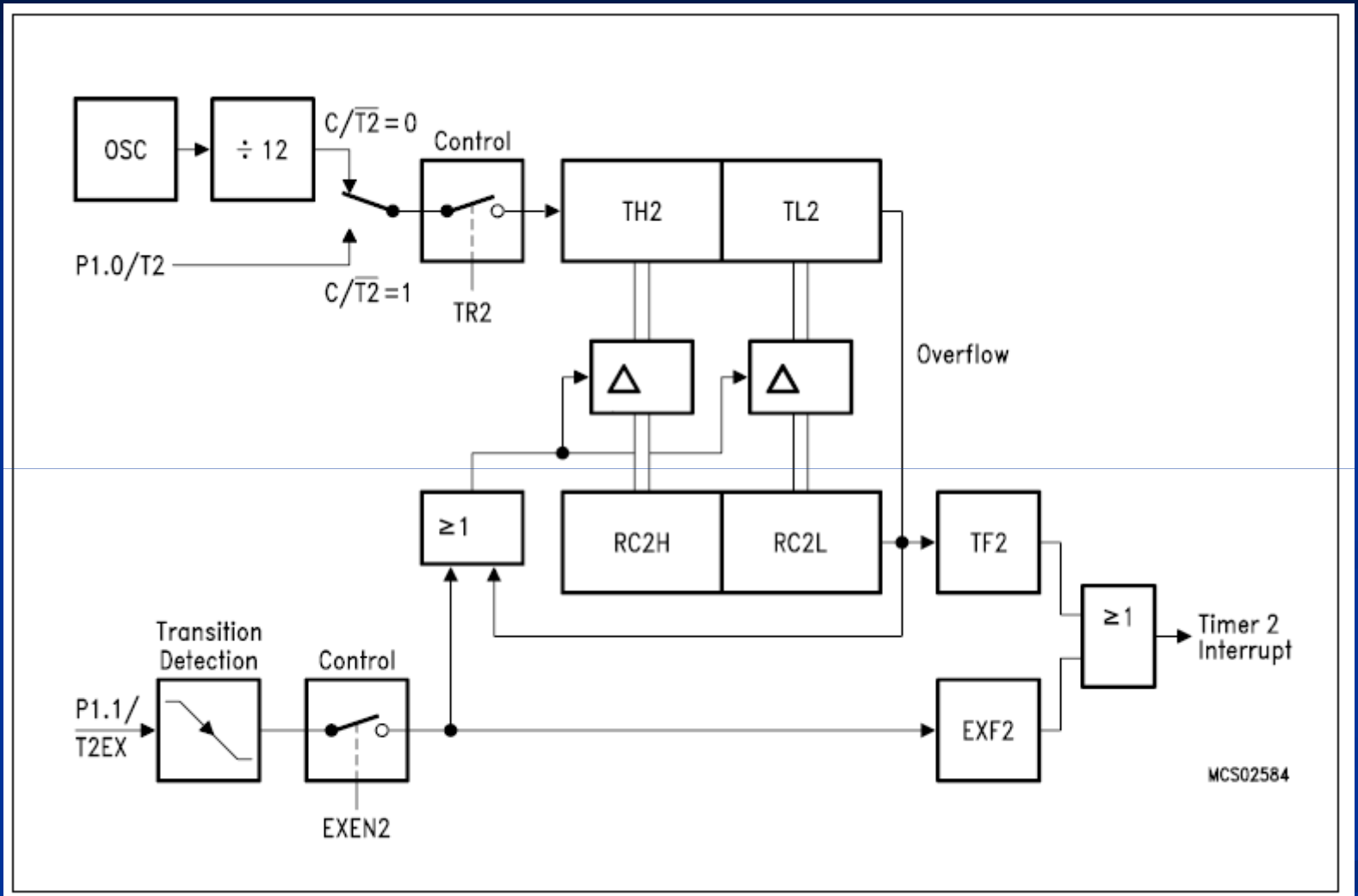
Special Function Register T2CON (Address $C8_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	$C/\overline{T2}$	$CP/\overline{RL2}$	T2CON

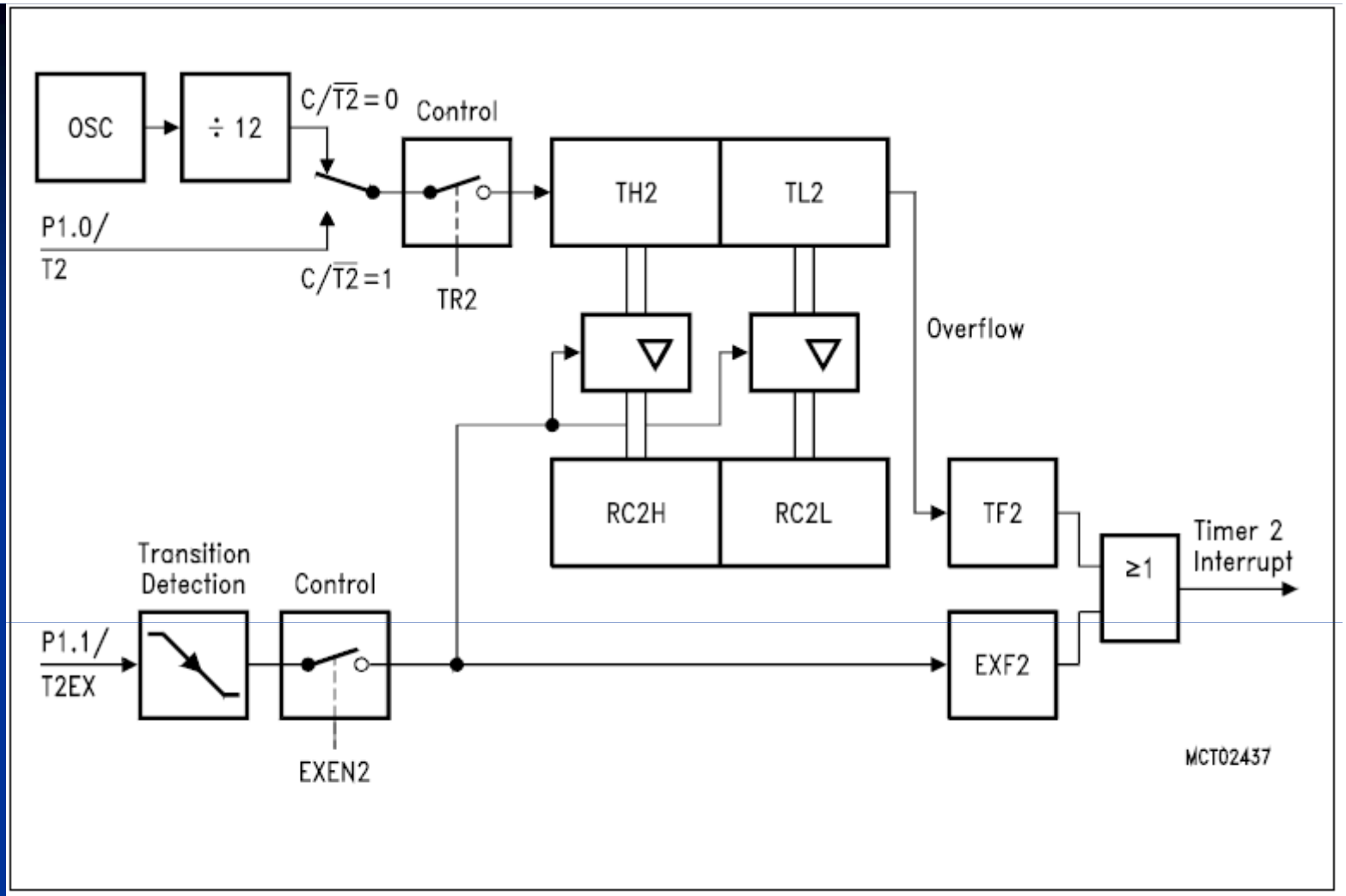
Bit	Function
TF2	Timer 2 Overflow Flag Set by a timer 2 overflow. Must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 External Flag Set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1, SFR T2MOD)
RCLK	Receive Clock Enable When set, causes the serial port to use timer 2 overflow pulses for its receive clock in serial port modes 1 and 3. RCLK = 0 causes timer 1 overflows to be used for the receive clock.
TCLK	Transmit Clock Enable When set, causes the serial port to use timer 2 overflow pulses for its transmit clock in serial port modes 1 and 3. TCLK = 0 causes timer 1 overflow to be used for the transmit clock.
EXEN2	Timer 2 External Enable When set, allows a capture or reload to occur as a result of a negative transition on pin T2EX (P1.1) if timer 2 is not being used to clock the serial port. EXEN2 = 0 causes timer 2 to ignore events at T2EX.
TR2	Start / Stop Control for Timer 2 TR2 = 1 starts timer 2.
$C/\overline{T2}$	Timer or Counter Select for Timer 2 $C/\overline{T2}$ = 0 for timer function. $C/\overline{T2}$ = 1 for external event counter (falling edge triggered).
$CP/\overline{RL2}$	Capture /Reload Select $CP/\overline{RL2}$ = 1 causes captures to occur an negative transitions at pin T2EX if EXEN2 = 1. $CP/\overline{RL2}$ = 0 causes automatic reloads to occur when timer 2 overflows or negative transitions occur at pin T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on timer 2 overflow.

RCLK + TCLK	$CP/\overline{RL2}$	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

TIMER mode AUTO-RELOAD



CAPTURE



Lors de l'événement (front montant ou descendant) sur une broche le contenu du TIMER 2 est recopié dans un registre, une interruption peut être générée. Le microcontrôleur conserve ainsi une trace très précise de l'instant ou s'est produit l'événement. Le traitement et les actions correspondantes peuvent être traitées ensuite.