Communication des Systèmes Embarqués

Bus de communication série et protocoles couches basses

yasmina.layouni@cpe.fr
damien.favre@cpe.fr

CPE LYON
ÉCOLE SUPÉRIEURE DE CHIMIE PHYSIQUE ÉLECTRONIQUE DE LYON

Ressources e-campus

- Cours
- Documentation architecture 8051
- Documentation architecture STM32
- Documentations techniques des capteurs
- **Liens pour déposer les codes source de TP et un rapport (compte-rendu) sous forme de note d'application**

Planning

Mardi 22/09 :

- 8h-10h : Cours sur l'architecture ARM ?
- 10h-12h : Cours sur les liaisons série
- 13h30-17h30 : Début de mise en pratique sur 20 heures
  - Mise en place liaison série sur 8051, communication avec un PC
  - Interfaçage accéléromètre. Envoi des données au PC
  - Ajout d'une seconde liaison SPI entre deux 8051
  - Affichage des données sur écran connecté au STM32
  - Interfaçage de l'accéléromètre directement sur le STM32

Mercredi 23/09 : Mise en pratique, suite

- 8h-12h : séance en semi-autonomie
- 13h30-17h30 : séance encadrée

Planning

Jeudi 24/09 :

- 13h30-17h30 : séance encadrée

Vendredi 25/09 :

- 8h-12h : séance encadrée, fin de la mise en pratique.
- **12h : RENDU E-CAMPUS. Fin du module CSE**
- 13h30-17h30 : Présentation et début du mini-projet

Vendredi 2/10

- Date limite de dépôt des rapports sous forme de note d'application

Liaison et bus série

INTRODUCTION

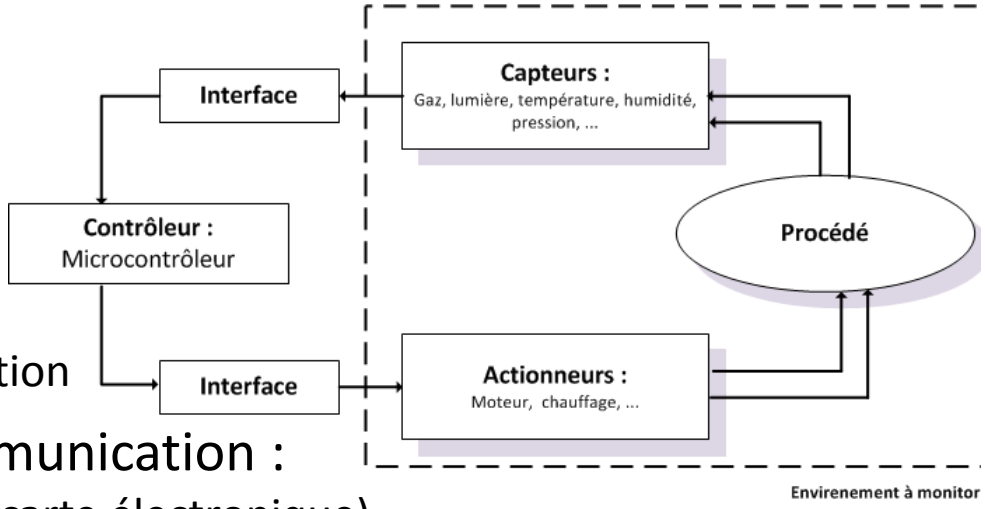
Introduction : liaison et bus série

1- Besoin : faire communiquer

- ✓ Capteurs
- ✓ Actionneurs
- ✓ Microcontrôleurs

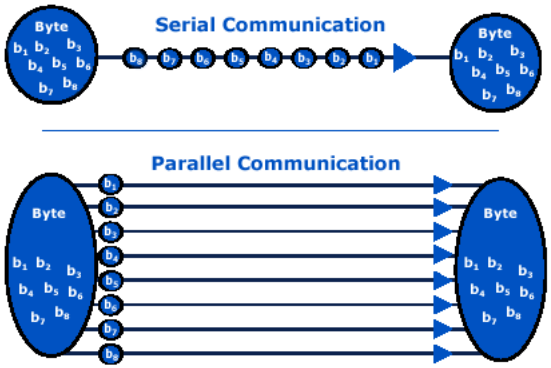
2- Nécessité : uniformiser

- ✓ Des standards définissent le support physique et le protocole de communication



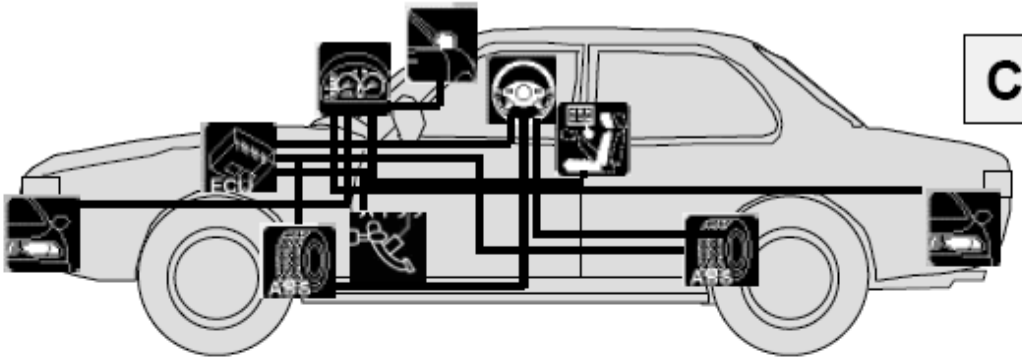
➢ Caractéristiques d'un bus de communication :

- Domaine d'application (bâtiment ou carte électronique)
- Nombre de périphériques (ou nœuds)
- Possibilité d'être maître ou esclave
- Débit de données
- Direction des données, half-duplex ou full-duplex
- Synchrone ou asynchrone
- Distance maximale
- Nombre de fils (hors alimentation et masse)



Introduction : liaison et bus série

Bus de terrain dans l'automobile : CAN

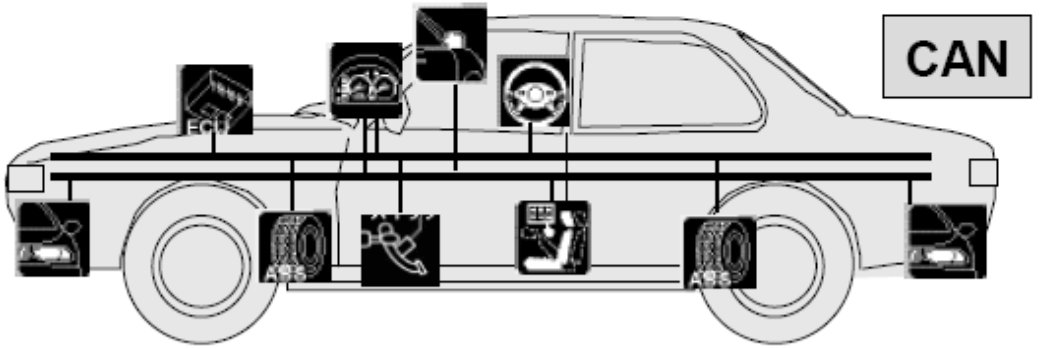


Câblage traditionnel

BUS CAN (Controller Area Network) : industrie, automobile

- Bus série multiplexé.
- Fonctionne en mode maître – esclave.

Longueur de câble : 2km



CAN

2 câbles pour l'alimentation électrique 12 V DC

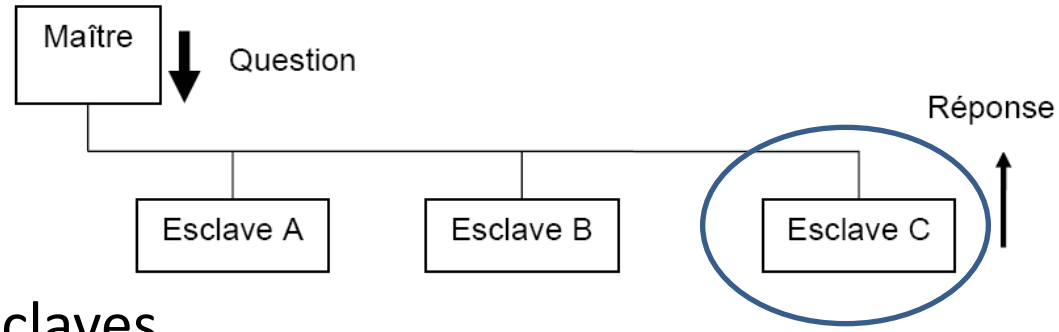
1 paire torsadée pour les transferts d'informations

Introduction : communication maître-esclave

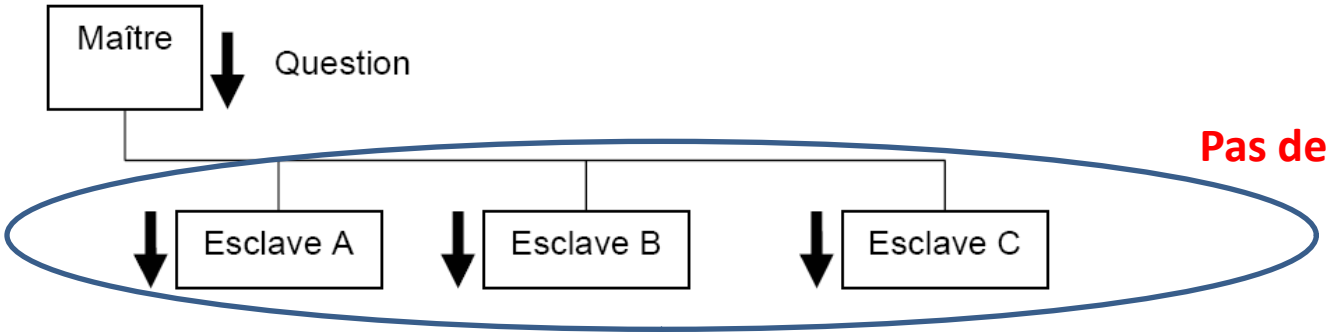
- Dans une communication maître-esclave, le maître choisit l'esclave avec lequel il communique :
  - Soit chaque esclave a un adresse unique (I²C)
  - Soit une ligne de communication par esclave permet de l'activer (SPI)
- Pas de communication entre les esclaves.

Deux types d'échanges :

- un maître -> un esclave



- Un maître -> plusieurs esclaves



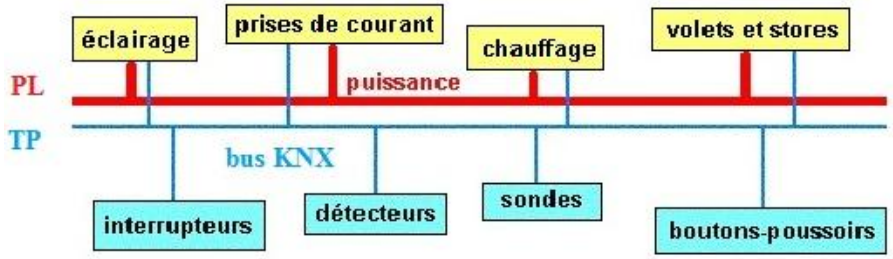
Pas de réponse possible

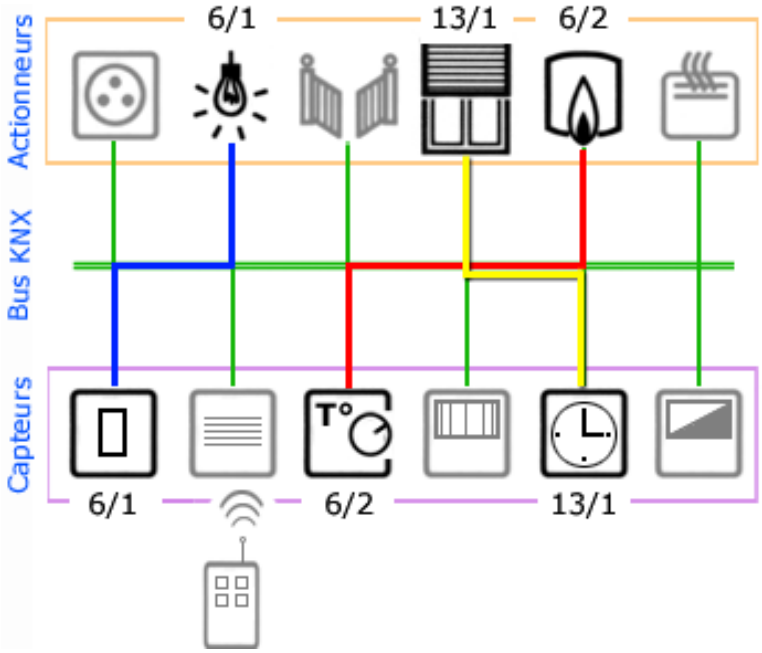
Introduction : liaison et bus série

Bus de terrain dans le bâtiment : KNX ou Konnex

- 3 supports physiques :
  - PL (Power Line : 230/400V 50 Hz)
  - TP (Twisted Pair ou paire torsadé : 21V-29V)
  - RF (liaison radio, porteuse à 868MHz)

Exemple avec le support TP :





Domotique

- ✓ Capteur de luminosité
  - ➢ Actionneur : store, lampe…
- ✓ Capteur de température
  - ➢ Actionneur : store, chaudière…

Introduction : liaison et bus série

Liaison série asynchrone

- Domaine d'application : carte électronique, inter-cartes
- 2 maîtres (chacun peut parler à l'autre quand il le souhaite)
- Débit moyen (10kbits/s à 250kbits/s)
- Liaison bidirectionnelle, full-duplex, asynchrone
- Courte distance (quelques dizaines de cm)
- Implémentation bas coût et mise en œuvre simple. Seulement deux lignes pour des performances très correctes
- **On peut ajouter un contrôle de flux**
- Structure d'une trame :
  - 1 bit de Start
  - 5-9 bits de données
  - 1 bit de parité optionnel
  - 1 ou 2 bits de Stop
- Tension de fonctionnement quelconque mais identique pour les deux périphériques (5V, 3,3V, 1,8V…)

Introduction : liaison et bus série

Liaison RS232 (1981)

Liaison série asynchrone avec des niveaux de tension adaptés à la moyenne distance (50m si faible débit)

- Domaine d'application : ordinateur et périphérique
- Le standard RS232 normalise la couche physique (connecteur, niveaux de tension, rôle de chaque broche) **mais pas le protocole**
- Connecteur SUB-D mâle à 9 points

- Tension de fonctionnement :
  - Maximale : ±25V
  - Souvent rencontrée : ± 12V
  - Minimale : ± 3Volts

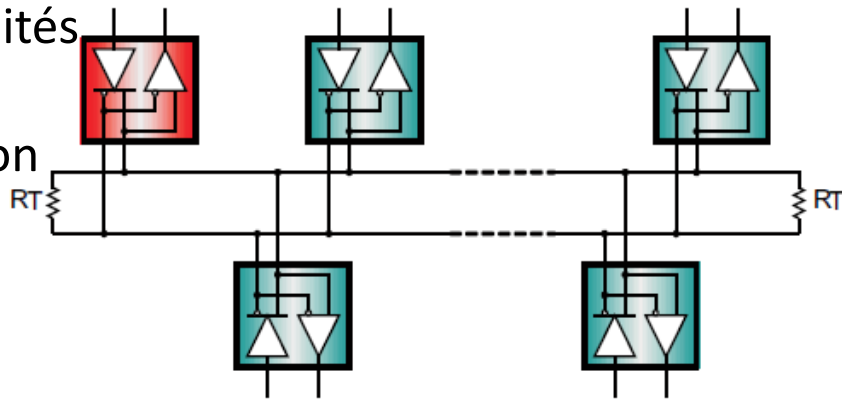
- **Sensible au bruit car le signal est référencé par rapport à la masse.**

Introduction : liaison et bus série

Liaison RS485 (1983)

Liaison série asynchrone multipoints avec des niveaux de tension adaptés à la longue distance et haut débit (jusqu'à 10Mbits/s *OU* 1km)

- Domaine d'application : bus de terrain
- Tension en mode différentiel par paire torsadée : apporte une immunité contre le bruit.
- Un émetteur peut communiquer avec 32 unités
- Au delà de 32, il faut utiliser des répéteurs
- Le bus a besoin de résistances de terminaison



RS485 n'impose pas de protocole, tout comme RS232.

Dans l'industrie, le protocole MODBUS est souvent utilisé sur la couche physique RS485.

Introduction : liaison et bus série

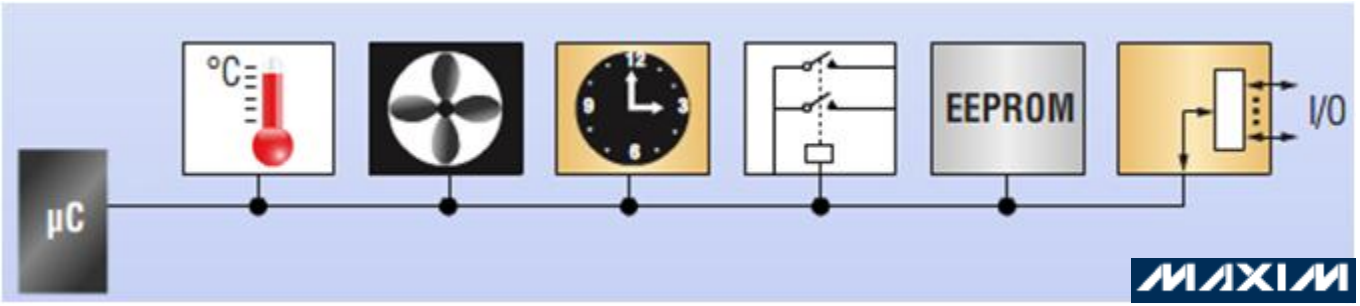
1-WIRE de Dallas (racheté par Maxim)

- Domaine d'application : carte électronique, bus de terrain peu perturbé
- 1 maître, plusieurs esclaves (le nombre max dépend de l'environnement)
- Faible débit (16kbits/s)
- Liaison bidirectionnelle, asynchrone, half-duplex
- Moyenne distance : plusieurs dizaines de mètres en environnement peu perturbé
- 1 ligne de donnée
- **Ligne pilotée par des drains (ou collecteurs) ouverts**
  - Création d'un OU logique
- Implémentation bas coût
  - Chaque composant a une seule adresse programmée en usine
  - Mise en œuvre simple (1 fil)

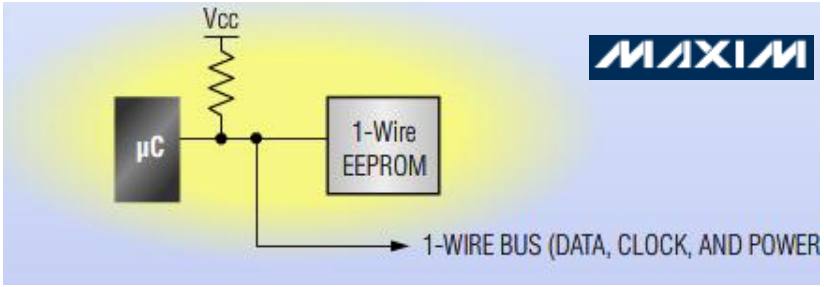
Le bus 1-WIRE

Le bus 1-WIRE a été développé par Dallas Semiconductor. Il permet de faire communiquer plusieurs circuits sur un seul fil.

- On trouve sur le bus **un seul maître** qui peut dialoguer avec un ou plusieurs esclaves.

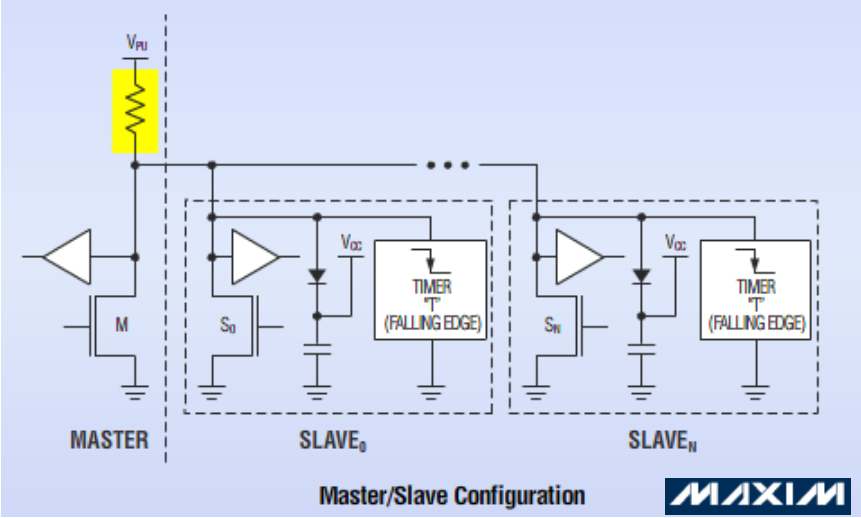


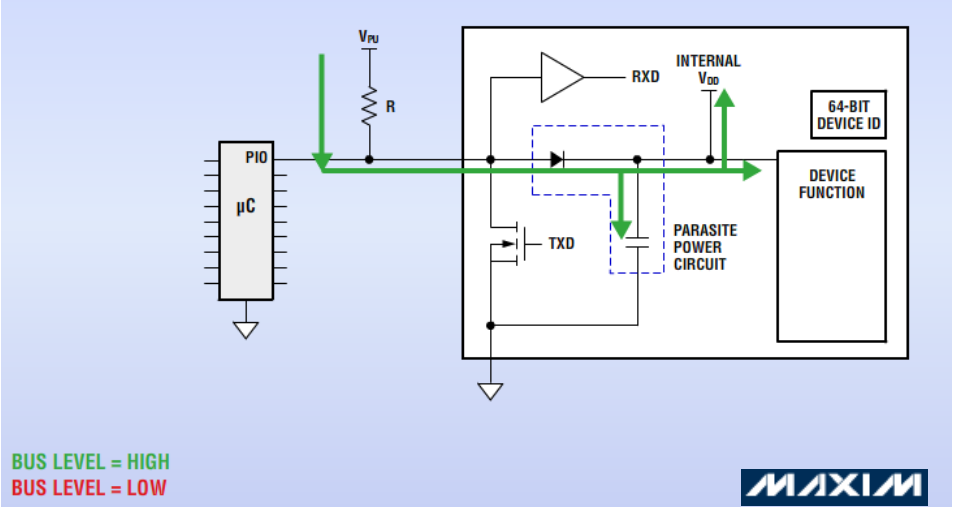
- Le bus de données est sur **un collecteur ouvert** polarisé via une résistance pull-up (1-5kΩ) à Vcc.
- L'état de repos du bus est **donc l'état haut**.



Le bus 1-WIRE

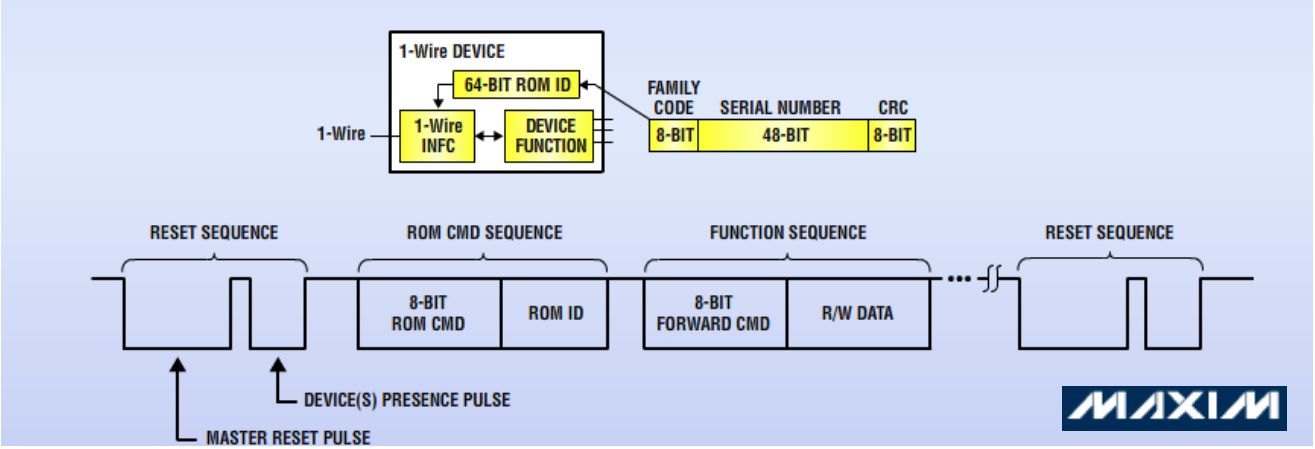
- Faible vitesse de communication (fixée à 16kbits/s).
- Bus faible coût, plusieurs esclaves sur un seul fil.
- Moyenne distance, jusqu'à 100m en conditions parfaites.
- Possibilité d'alimenter les différents esclaves à partir de la ligne de communication (mode parasite)**. Les capteurs peuvent ainsi être connectés au système par seulement deux fils (signal et masse)**.





Le bus 1-WIRE

- Chaque esclave possède une adresse physique **unique** de 64 bits (gravée en dur sur la puce lors de la fabrication).



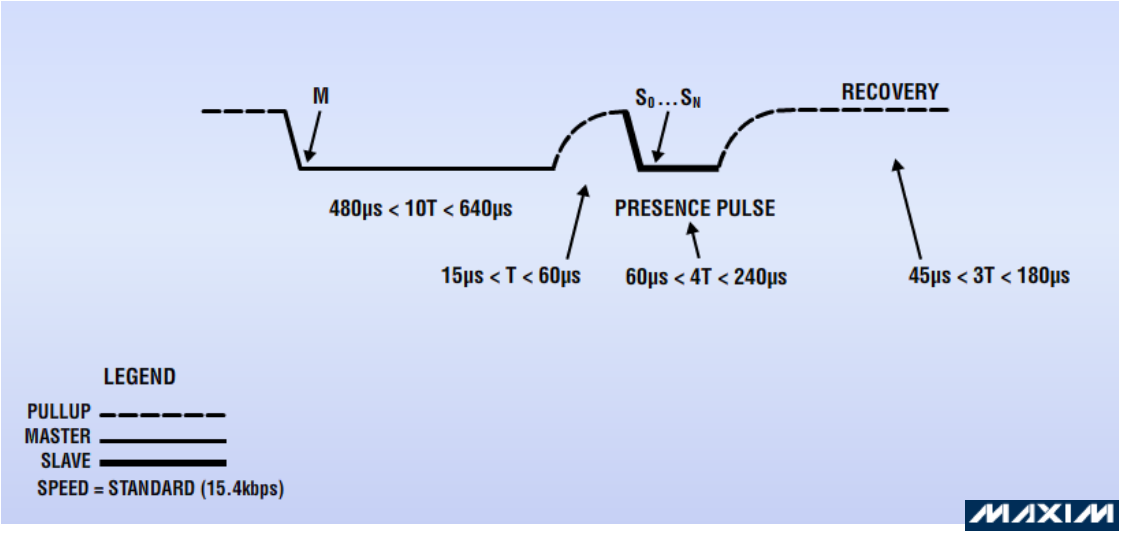
- Avant toute communication, le maître réinitialise les composants connectés.
- Le maître peut alors identifier les esclaves présents.

Dallas a un monopole de l'allocation d'adresses. Seule l'utilisation de ses produits garantit qu'il n'y aura pas deux adresses identiques sur le bus.

Le bus 1-WIRE

Protocole :

1. Le maître met le bus à « 0 » pendant au moins 480µs pour réinitialiser les composants connectés.
2. Les esclaves répondent alors par une « impulsion de présence » : le maître sait alors si au moins un esclave est présent.



Introduction : liaison et bus série

I²C de Philips (ou SMBUS) : *Inter Integrated Circuit*

- Domaine d'application : carte électronique, inter-cartes
- Multi-maîtres, multi-esclaves (jusqu'à 127)
- Débit 100kbit/s (standard), 400kbit/s (fast)
- Liaison bidirectionnelle, synchrone, half-duplex
- Courte distance (environ 1m)
- 2 lignes (SDA, SCL)
- **Lignes pilotées par des drains (ou collecteurs) ouverts**
  - Création d'un OU logique
- Notion d'acquiescement par le récepteur
- Conçu à l'origine pour faire communiquer les différents composants d'une carte, notamment dans les téléviseurs

Introduction : liaison et bus série

SPI de Motorola : *Serial Peripheral Interface*

- Domaine d'application : carte électronique
- 1 maître, nombre limité d'esclaves (1-5)
- Débit important, plusieurs Mbps voire dizaines de Mbps
- Liaison synchrone, bidirectionnelle, full-duplex
- Courte distance (dizaine de cm)
- 4 lignes (MOSI, MISO, SCK, SS)
- Mise en œuvre un peu plus complexe
  - Une ligne par esclave nécessaire pour sa sélection : devient gourmand en I/O lorsque le nombre d'esclaves augmente.

Application : lecture de données sur liaisons série

Réalisation d'un système de surveillance des conditions de transport de marchandises. On veut détecter d'éventuels chocs grâce à un accéléromètre 3 axes ADXL345 qui utilise une liaison SPI.

- Les données issues de ce capteur sont dans un premier temps envoyées à un PC via une liaison RS232.
- Dans un second temps, elles seront affichés sur un écran LCD.
- Ensuite, on ajoutera un microcontrôleur intermédiaire communiquant via une liaison SPI.

Ports I/O

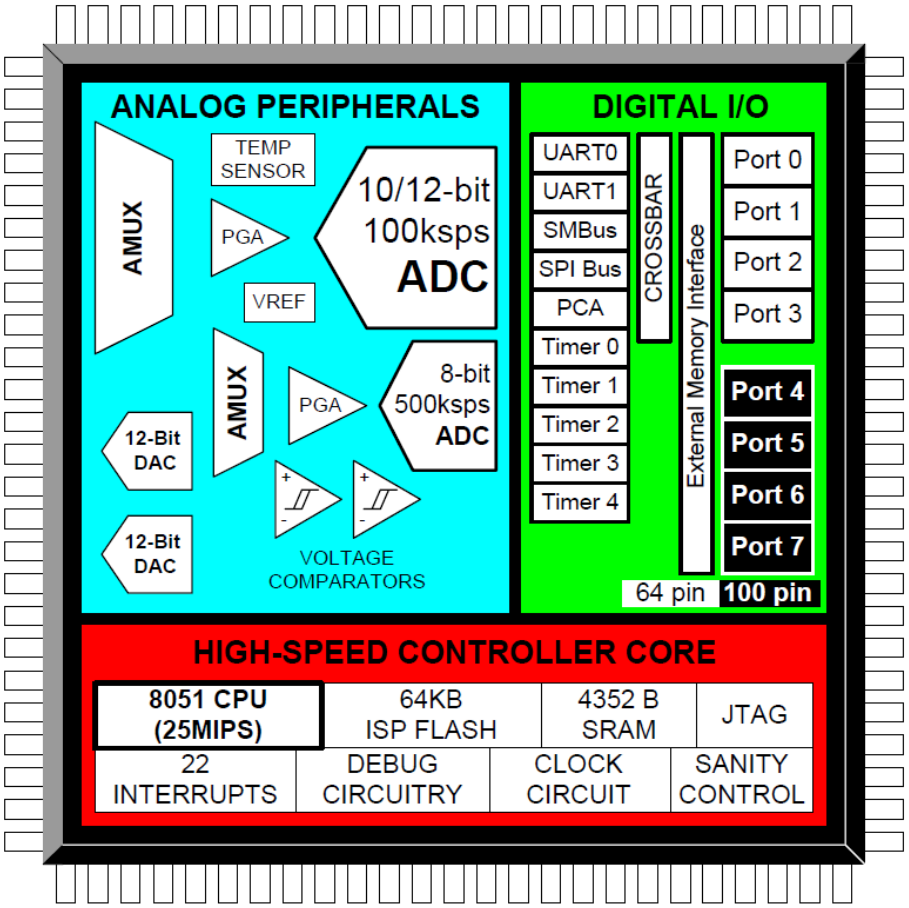
PUSH PULL/OPEN DRAIN
CROSSBAR DU µC 8051

Le microcontrôleur 8051 et ses périphériques

Le cœur 8051 a été développé par Intel au début des années 1980.

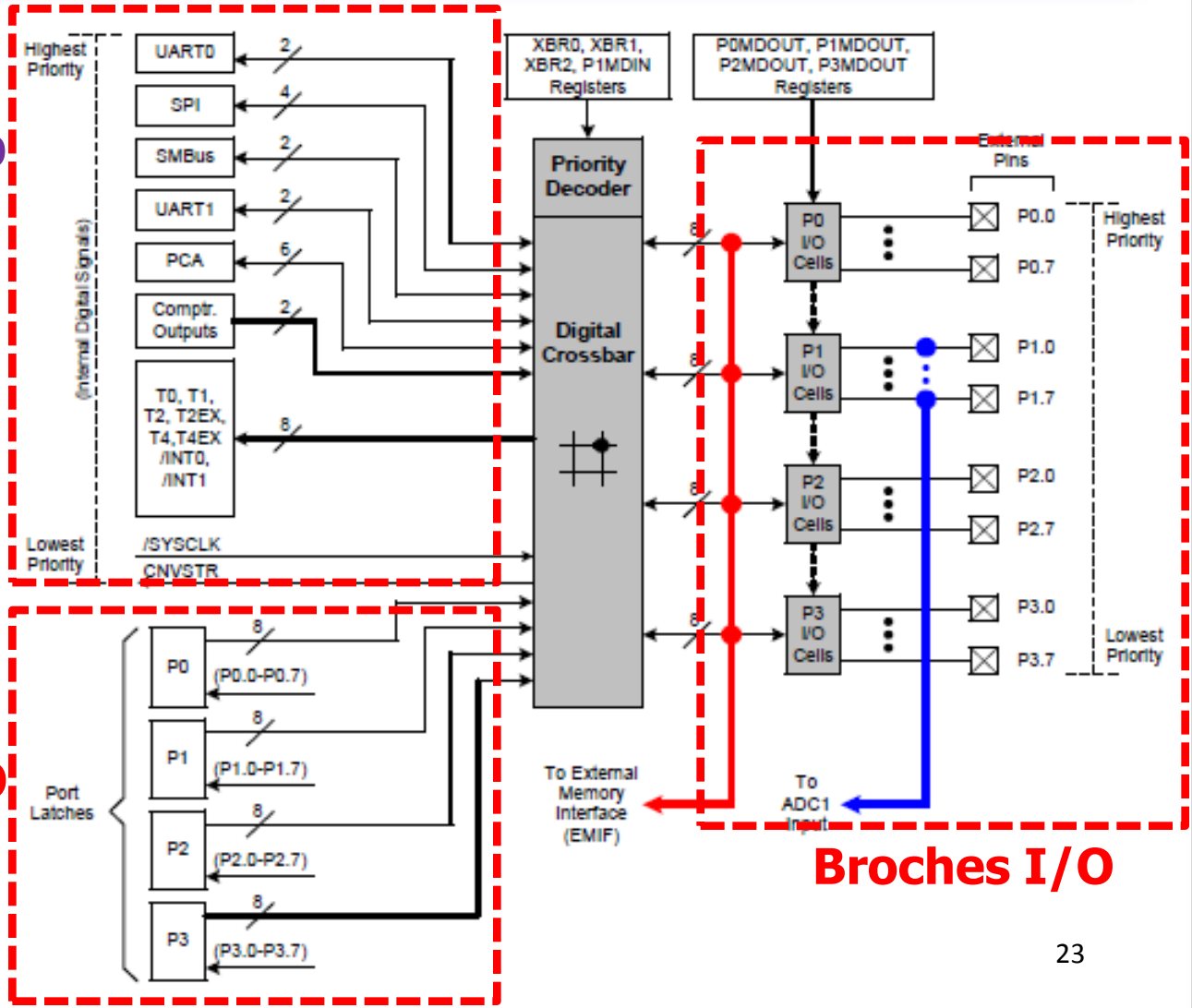
Le CPU original n'est plus utilisé mais de nombreux microcontrôleurs plus modernes embarquent un cœur 8051.

C'est une architecture ancienne mais éprouvée, encore utilisée dans de nombreux domaines où la fiabilité doit être excellente ou le coût extrêmement bas.



Ports I/O et crossbar

Crossbar : « aiguilleur » entre les signaux internes du 8051 et les broches I/O



Registres :
XBR0, XBR1 et XBR2

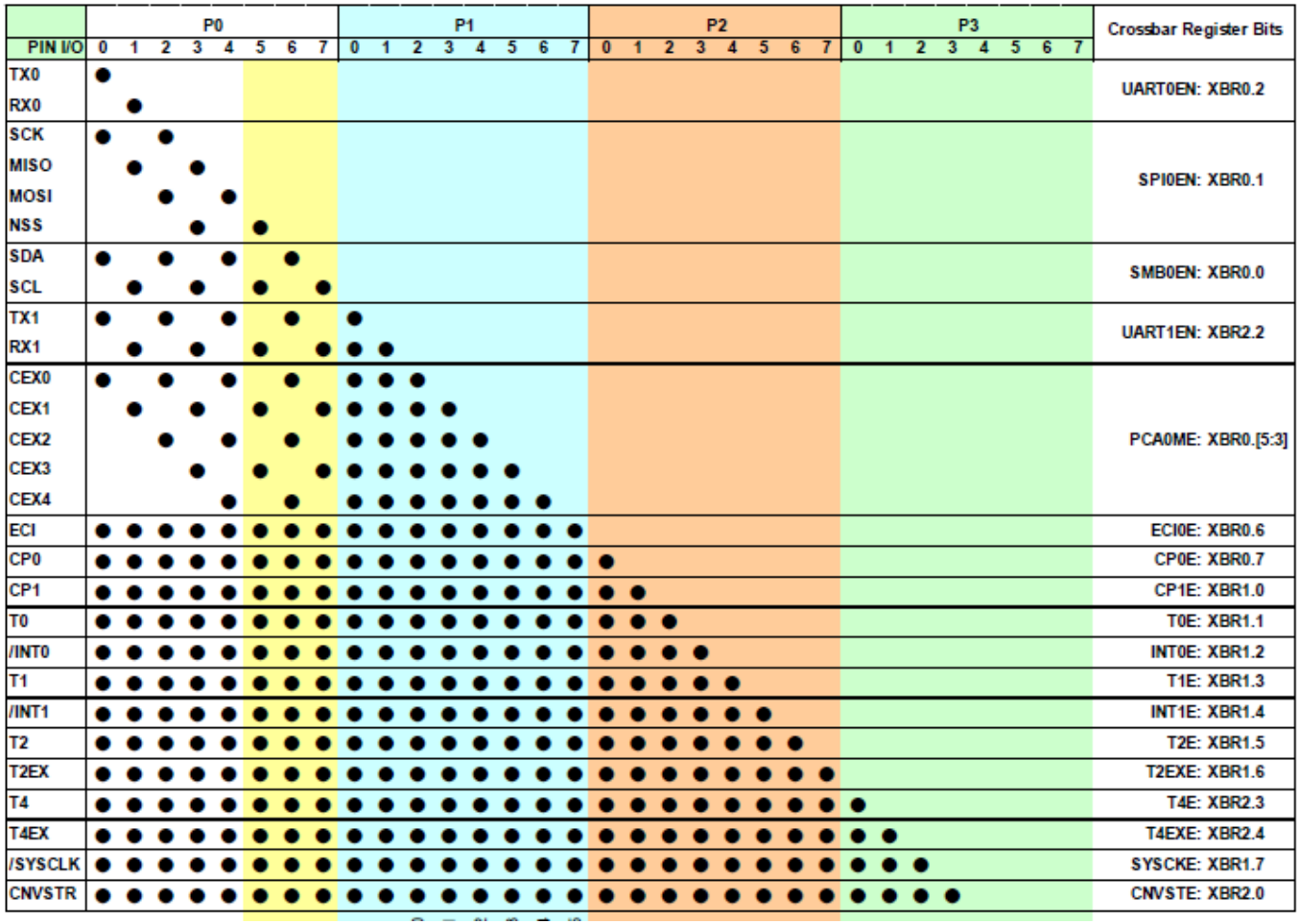
Configuration avec GUI grâce à l'outil *Silicon Labs Wizard*

Ports I/O et crossbar

Configuration du crossbar avec l'outil graphique :

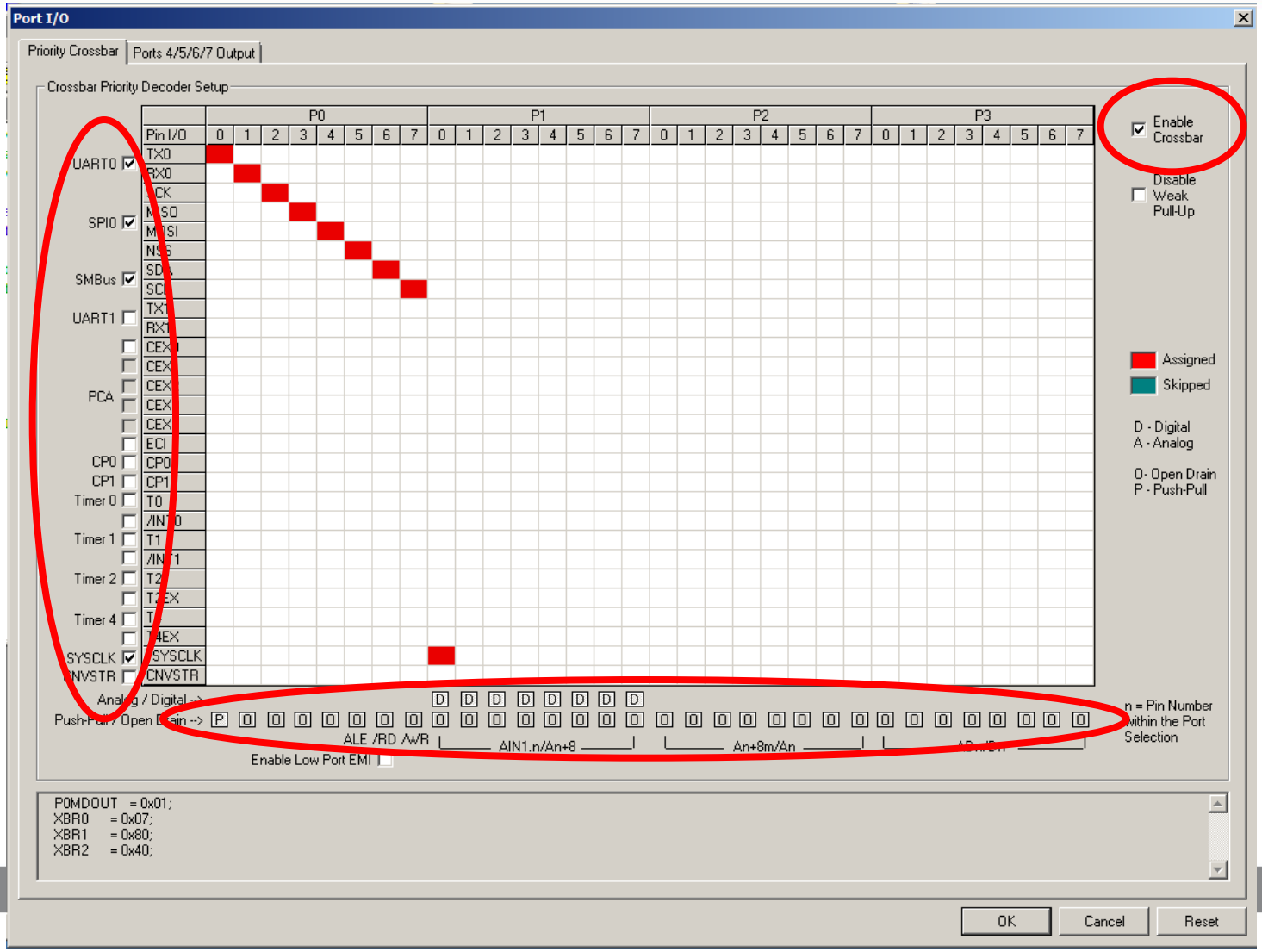
Le périphérique le plus prioritaire est le plus haut sur le schéma.

Chaque périphérique activé réquisitionne des ports I/O en prenant les plus à gauche du schéma.



Ports I/O et crossbar

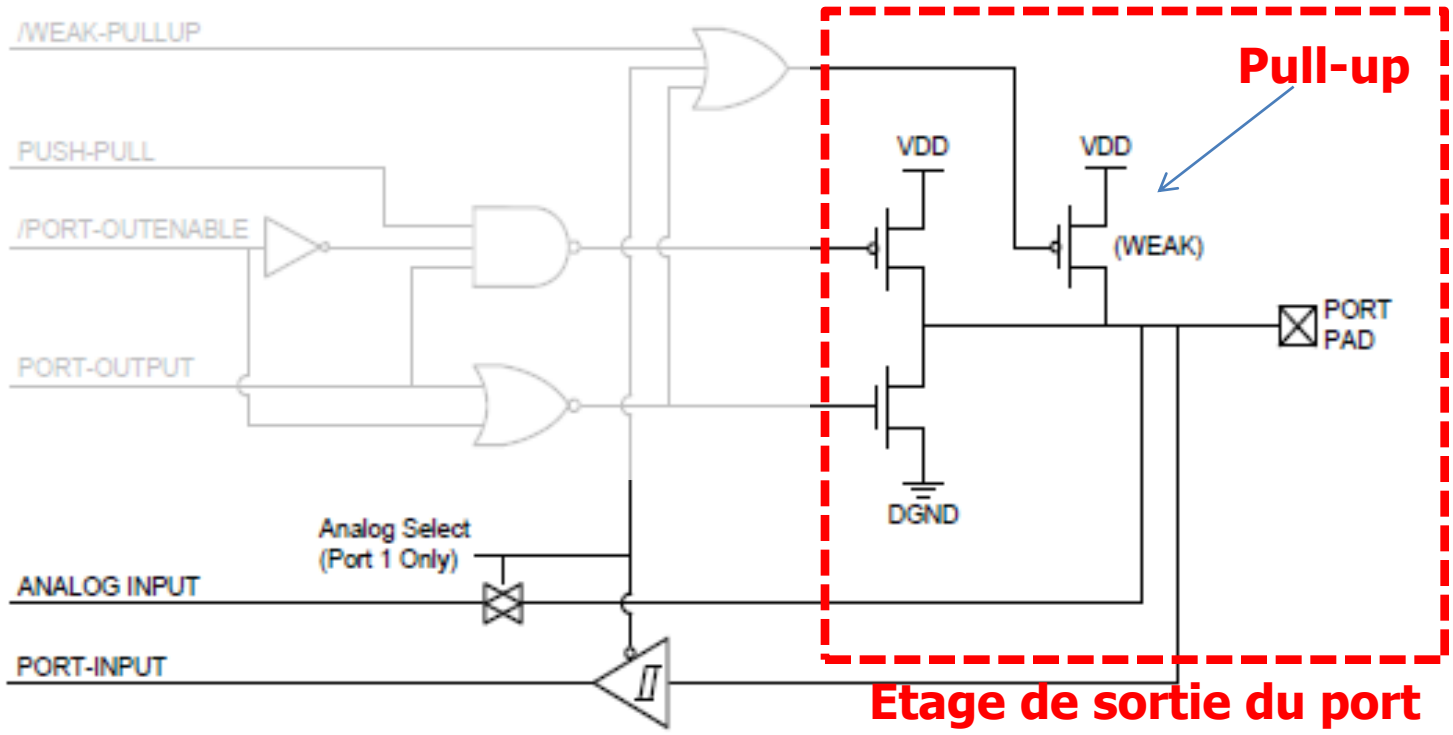
Outil de configuration « Wizard »



Ports I/O : push-pull VS drain ouvert

Ports I/O :

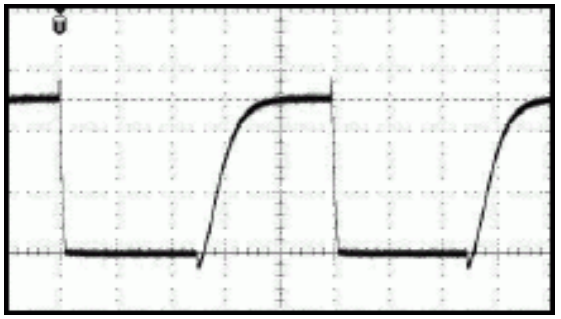
- Choix possibles entre les modes **push-pull, drain ouvert (avec ou sans pull-up) et analogique**
- Registres : **PxMDOUT, PxMDIN, Px.n**



Ports I/O : push-pull VS drain ouvert

- Port utilisé en sortie:
  - Le mode push-pull (PP) permet d'avoir des signaux rapides
  - Le mode open drain (OD) permet de réaliser un « OU » câblé.

    → 1-Wire, I^2C.
  - **Son principal inconvénient provient de la lenteur de la remontée du signal :**



- Port utilisé en entrée:
  - Configurer le port en drain ouvert pour bloquer le transistor « du haut ».
  - Ecrire un 1 dans le registre de sortie du port pour bloquer le transistor « du bas »

Ports I/O : configuration en entrée

Figure 17.14. P1MDOUT: Port1 Output Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xA5

Bits7-0: P1MDOUT.[7:0]: Port1 Output Mode Bits.
0: Port Pin output mode is configured as Open-Drain.
1: Port Pin output mode is configured as Push-Pull.

Note: SDA, SCL, and RX0 (when UART0 is in Mode 0) and RX1 (when UART1 is in Mode 0) are always configured as Open-Drain when they appear on Port pins.

```
P1MDOUT &= ~0x40; // met P1.6 en drain ouvert
```

Ports I/O : configuration en entrée

Le port a été mis en drain ouvert, il reste à écrire un 1 dans son registre de sortie :

Figure 17.12. P1: Port1 Data Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (bit addressable)	SFR Address: 0x90

Bits7-0: P1.[7:0]: Port1 Output Latch Bits.
(Write - Output appears on I/O pins per XBR0, XBR1, XBR2, and XBR3 Registers)
0: Logic Low Output.
1: Logic High Output (open if corresponding P1MDOUT.n bit = 0).
(Read - Regardless of XBR0, XBR1, XBR2, and XBR3 Register settings).
0: P1.n pin is logic low.
1: P1.n pin is logic high.

```
sbit mon_entree= P1^6;
...
mon_entree = 1;
```

Ports I/O : configuration en sortie

Figure 17.14. P1MDOUT: Port1 Output Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xA5

Bits7-0: P1MDOUT.[7:0]: Port1 Output Mode Bits.
0: Port Pin output mode is configured as Open-Drain.
1: Port Pin output mode is configured as Push-Pull.

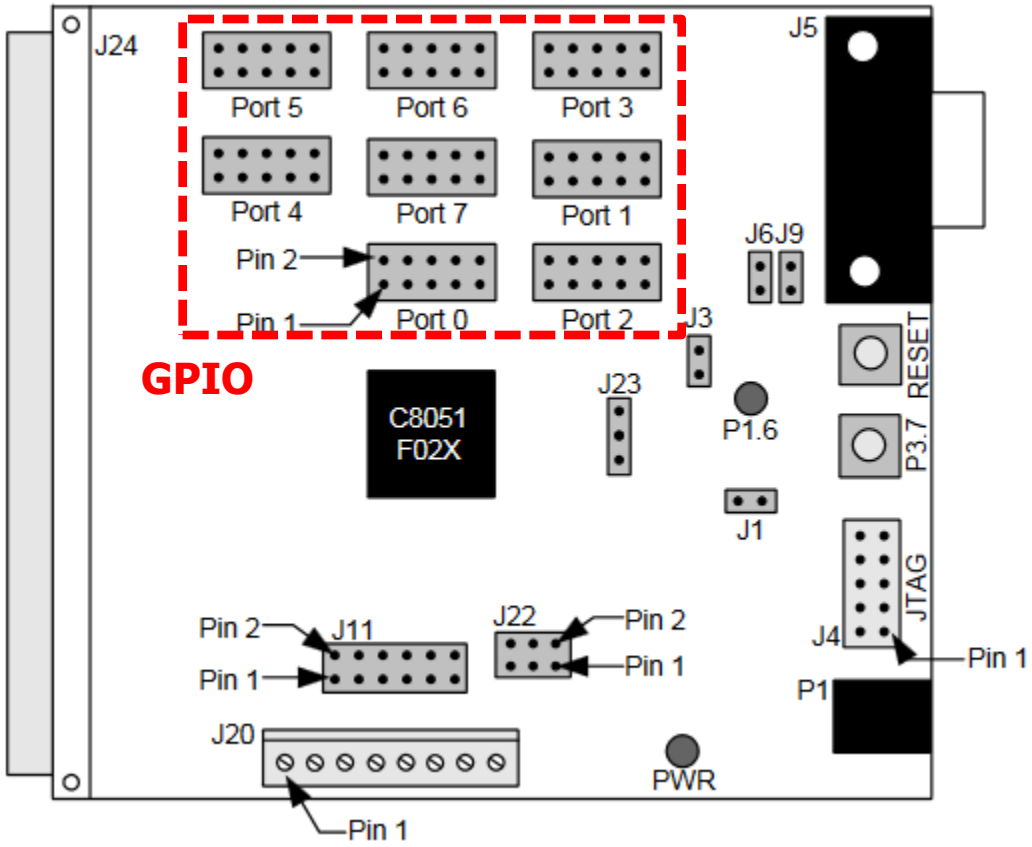
Note: SDA, SCL, and RX0 (when UART0 is in Mode 0) and RX1 (when UART1 is in Mode 0) are always configured as Open-Drain when they appear on Port pins.

Après le reset du microcontrôleur (µC) : les ports sont en drain ouvert

```
P1MDOUT |= 0x01; // met P1.0 en push-pull
```

```
P1 &= ~0x01; // sortie P1.0 à l'état bas
P1 |= 0x01; // sortie P1.0 à l'état haut
```

Ports I/O : câblage sur la carte d'évaluation



Pin #	Description
1	Pn.0
2	Pn.1
3	Pn.2
4	Pn.3
5	Pn.4
6	Pn.5
7	Pn.6
8	Pn.7
9	+3 VD (+3.3 VDC)
10	GND (Ground)

Timers

FONCTIONNEMENT
MISE EN ŒUVRE

Timers : rappel du fonctionnement

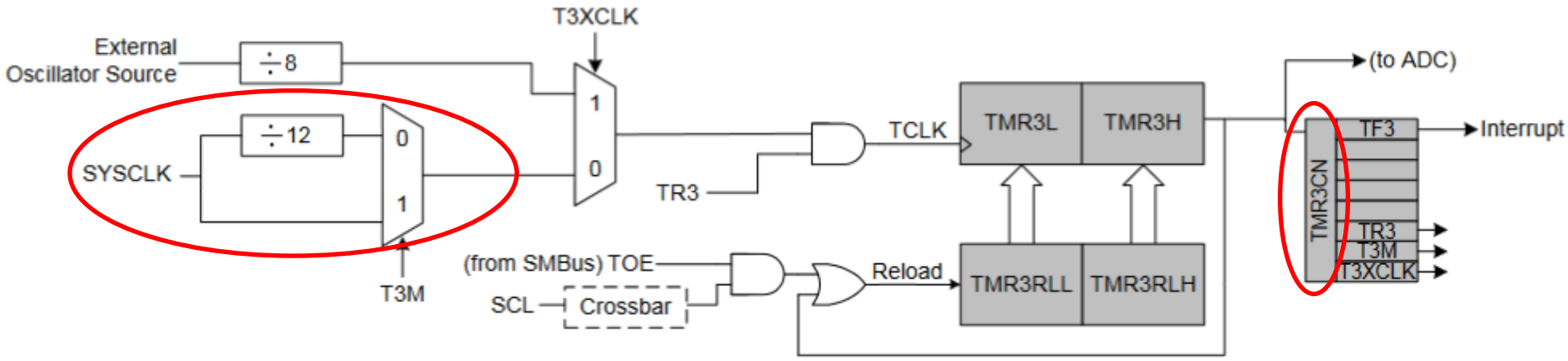
- Un timer est **cadencé** :
  - par un sous-multiple de l'horloge interne (le plus fréquent)
  - par un signal externe appliqué sur une broche I/O (plus rare).

- A chaque impulsion, sa valeur s'incrémente.

- Lorsqu'il atteint sa valeur max, un événement se produit : **débordement (*overflow*)**
  - Un timer basique sans fonction *auto-reload* retombe à 0
  - Un timer avec fonction *auto-reload* retombe à une valeur programmée
  - Une interruption est éventuellement lancée

Timers : configuration

Timer 0 and Timer 1:	Timer 2:	Timer 3:	Timer 4
13-bit counter/timer	16-bit counter/timer with auto-reload	16-bit timer with auto-reload	16-bit counter/timer with auto-reload
16-bit counter/timer	16-bit counter/timer with capture		16-bit counter/timer with capture
8-bit counter/timer with auto-reload	Baud rate generator for UART0		Baud rate generator for UART1
Two 8-bit counter/timers (Timer 0 only)			



Timers : configuration

Figure 22.20. TMR3CN: Timer 3 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF3	-	-	-	-	TR3	T3M	T3XCLK	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x91

Bit7: TF3: Timer3 Overflow Flag.
Set by hardware when Timer 3 overflows from 0xFFFF to 0x0000. When the Timer 3 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 3 Interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bits6-3: UNUSED. Read = 0000b, Write = don't care.

Bit2: TR3: Timer 3 Run Control.
This bit enables/disables Timer 3.
0: Timer 3 disabled.
1: Timer 3 enabled.

Bit1: T3M: Timer 3 Clock Select.
This bit controls the division of the system clock supplied to Counter/Timer 3.
0: Counter/Timer 3 uses the system clock divided by 12.
1: Counter/Timer 3 uses the system clock.

Bit0: T3XCLK: Timer 3 External Clock Select
This bit selects the external oscillator input divided by 8 as the Timer 3 clock source. When T3XCLK is logic 1, bit T3M (TMR3CN.1) is ignored.
0: Timer 3 clock source defined by bit T3M (TMR3CN.1).
1: Timer 3 clock source is the external oscillator input divided by 8.

Timers : configuration

Figure 12.12. EIE2: Extended Interrupt Enable 2

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

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

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

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

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

Bit3: EADC1: Enable ADC1 End Of Conversion Interrupt.
This bit sets the masking of the ADC1 End of Conversion interrupt.
0: Disable ADC1 End of Conversion interrupt.
1: Enable interrupt requests generated by the ADC1 End of Conversion Interrupt.

Bit2: ET4: Enable Timer 4 Interrupt
This bit sets the masking of the Timer 4 interrupt.
0: Disable Timer 4 interrupt.
1: Enable interrupt requests generated by the TF4 flag (T4CON.7).

Bit1: EADC0: Enable ADC0 End of Conversion Interrupt.
This bit sets the masking of the ADC0 End of Conversion Interrupt.
0: Disable ADC0 Conversion Interrupt.
1: Enable interrupt requests generated by the ADC0 Conversion Interrupt.

Bit0: ET3: Enable Timer 3 Interrupt.
This bit sets the masking of the Timer 3 interrupt.
0: Disable all Timer 3 interrupts.
1: Enable interrupt requests generated by the TF3 flag (TMR3CN.7).

Ne pas oublier d'activer aussi le système global d'interruption !

Timers : configuration

Calcul de la valeur de rechargement :

- On part de SYSCLK = 22,1184 Mhz (0,045µs)
- Pas de sous-division → incrémentation timer toutes les 0,045µs
- Durée souhaitée entre débordements : 15µs
- Nombre d'incréments nécessaires : durée souhaitée / durée 1 incrément
  - 15/0,045 = 333
  - TMR3RL = 65536-333
- **Durée maxi entre débordements = durée 1 incrément * 65536 = 2,9ms**
- **Si besoin d'une durée entre débordements > 2,9ms :**
  - **Utiliser un timer de + grande capacité (non existant sur le 8051)**
  - **Diviser la fréquence d'horloge source (possibilité par 12 ici)**

Liaison série asynchrone

UART / RS232

Liaison série asynchrone

Liaison série asynchrone

- Signaux de la liaison série
  - TX et RX : données
  - RTS, CTS, DSR, DCD et DTR : contrôle de flux (optionnel : permet de s'assurer que le correspondant est prêt à écouter avant d'émettre)

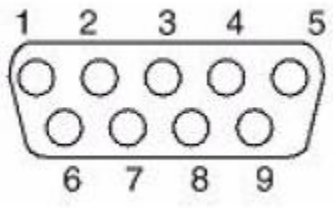
- UART : **Universal Asynchronous Receiver-Transmitter**
  - Liaison asynchrone : les données (5 à 9 bits) sont transmises sans horloge, entourées d'un bit de Start et d'un bit de Stop
  - Liaison full-duplex (pas de risque de collision) : émetteur et récepteur peuvent parler en même temps car chaque direction utilise un fil distinct.
  - Niveaux de tension : ceux utilisés par les E/S du microcontrôleur

- Norme RS232 :
  - Niveaux de tension -3/-15V (1 logique) et +3/+15V (0 logique),

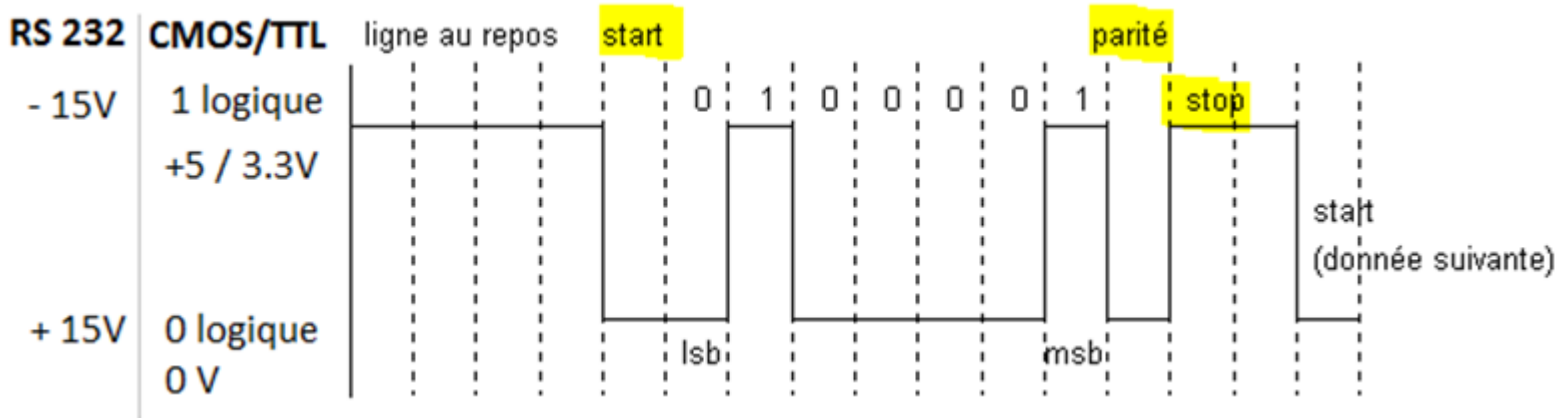
    → **logique inversée /!**

Liaison série asynchrone

- Connecteur DB9 : mâle côté PC
  - Broche 2 : RX
  - Broche 3 : TX
  - Broche 5 : GND



- Forme d'une trame série (niveaux RS232 et TTL/CMOS) :

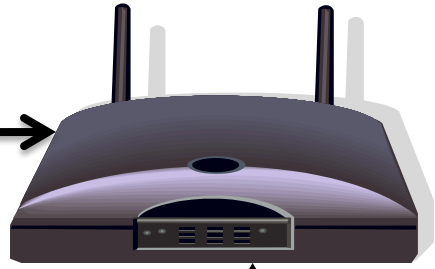


- Convertir le niveau logique (TX et RX) du microcontrôleur en niveau RS232 nécessite l'utilisation d'un circuit d'interfaçage.
  - **Exemple de circuit d'interface : le MAX232**

Liaison série asynchrone

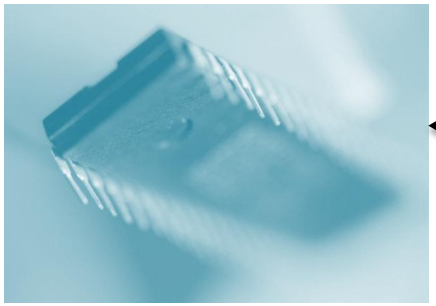
- Liaison UART/RS232

Connexion RS232
logique inversée, niveaux +/-15V



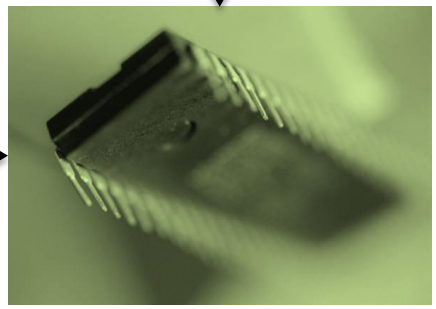
Interface RS232/
CMOS, TTL
Ajustement de niveaux

Interface RS232/CMOS, TTL
Ajustement de niveaux



UART

UART



Interface TTL / CMOS

/!\ Ajustement de niveau ou pas ?

Liaisons série

ADAPTATION DES NIVEAUX LOGIQUES

Adaptation des niveaux logiques

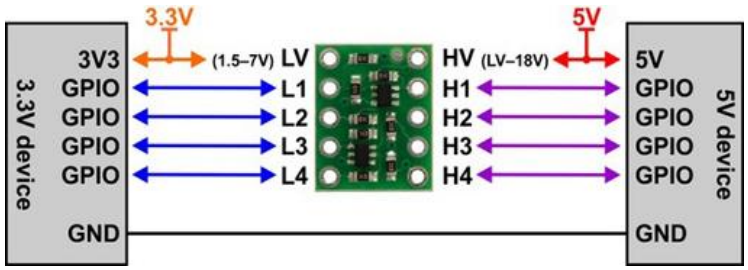
- Certains composants devant communiquer entre eux ne fonctionnent pas forcément avec les mêmes niveaux logiques :
  - RS232 : +/- 3 à 25V
  - TTL : 0/5V
  - 0/3,3V
  - 0/2,5V
  - 0/1,8V
  - 0/1,2V
  - etc.

Risque de destruction !

- **Puis-je connecter sans risques ces composants entre eux, si oui pourquoi, sinon comment faire ?**

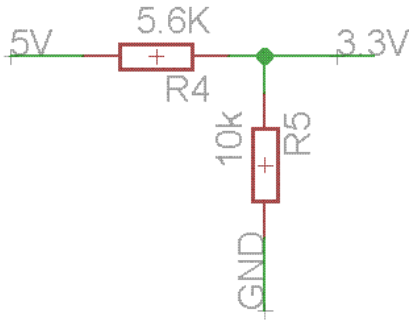
Adaptation des niveaux logiques

1. Utiliser un translateur de niveaux actif :



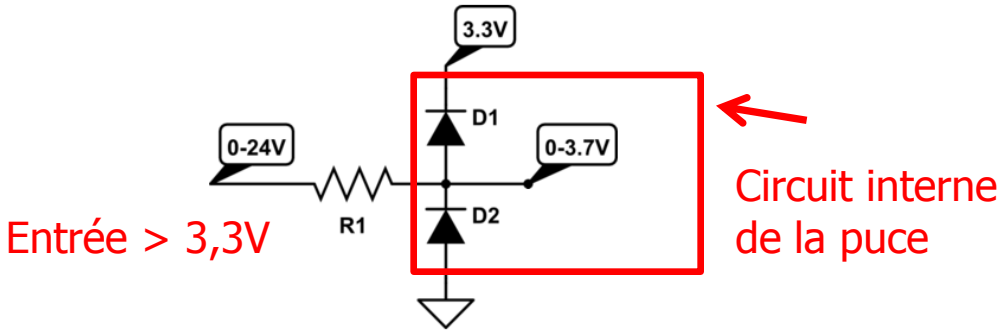
2. Branchement d'une entrée sur une sortie de tension supérieure :

Pont diviseur de tension :



Méthode sûre

mise à profit des diodes de protection ESD :



Nécessite des précautions, mais économise une résistance

Adaptation des niveaux logiques

3. Branchement d'une entrée sur une sortie de tension inférieure :

→ Connexion directe : Pas de risque de destruction. **Le niveau haut peut par contre ne pas être pris en compte.**

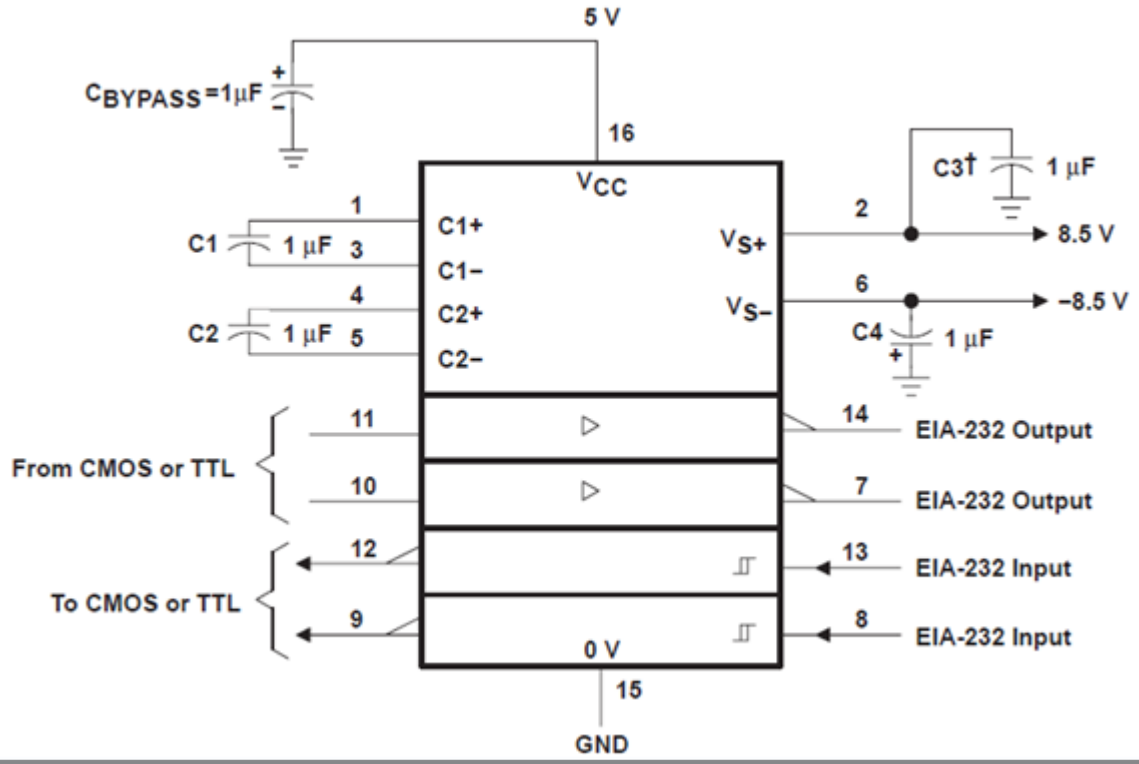
Pour que cela fonctionne à coup sûr, il faut **que $Voh_{min} > Vih_{min}$**

(tension minimale de la sortie à l'état haut > tension minimale pour qu'une entrée soit reconnue comme un état haut)

Si cela n'est pas le cas, le mieux est de passer par un translateur de niveau actif.

Adaptation des niveaux logiques

- Circuit d'interface pour des signaux compatibles RS232 (+/- 12V)
  - **MAX232** lorsque Valim = 5V
  - **MAX320** lorsque Valim = 3,3V



Liaison série asynchrone

MISE EN ŒUVRE SUR LE 8051

Mise en œuvre de la liaison série asynchrone

Configuration nécessaire du microcontrôleur 8051 :

- Configuration du cœur (horloge interne ou externe, reset, watchdog…)
- Crossbar (redirection sur des broches I/O des signaux utilisés par les périphériques)
- Port I/O : entrée ou sortie, push-pull ou drain ouvert ?
- Timer : pour cadencer l'UART
- Configuration de l'UART0 ou UART1 :
  - Mode : asynchrone, 8 bits, baud rate, parité ?
  - **Activation des interruptions ou pas ?**

Mise en œuvre de la liaison série asynchrone

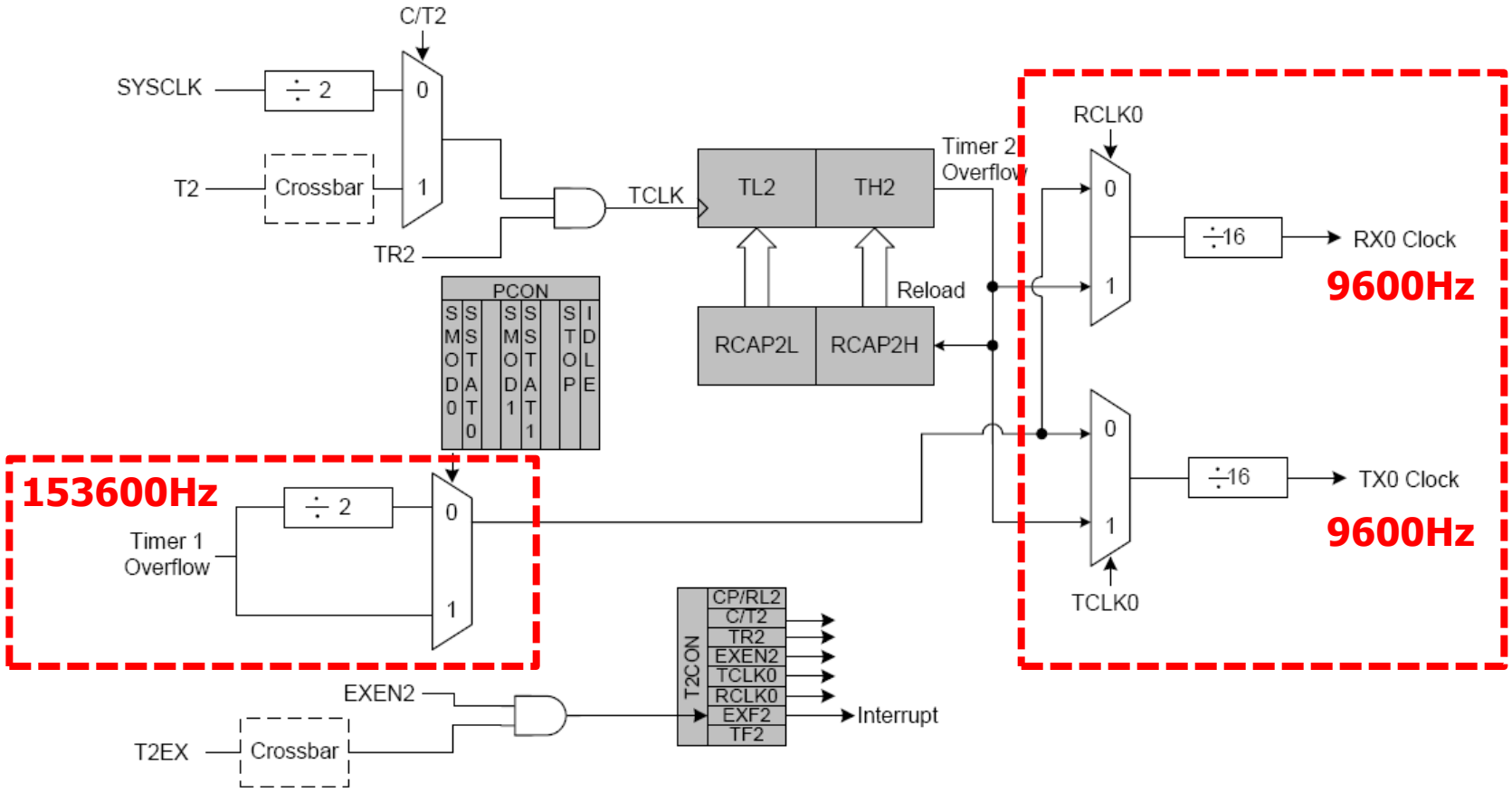
1. Générer une horloge pour cadencer l'UART0.
   - Choix possible Timer 1 ou Timer 2
   - Fréquence à générer par le timer : baudrate * 16 (l'UART a besoin d'une base de temps 16 fois plus rapide que le baudrate à générer).

2. Configurer l'UART proprement dit en mode 1 (asynchrone, 8 bits)

Table 20.1. UART0 Modes

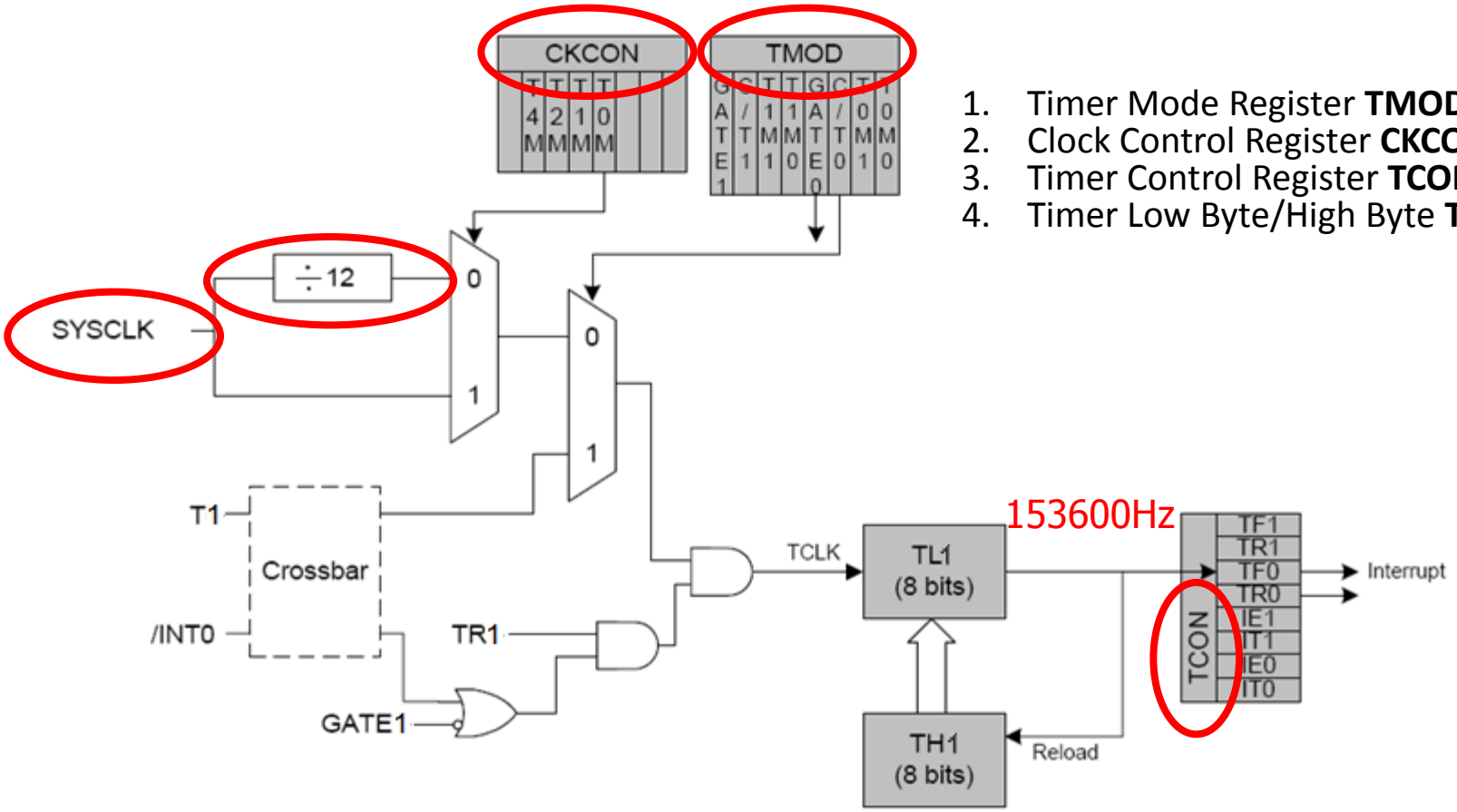
Mode	Synchronization	Baud Clock	Data Bits	Start/Stop Bits
0	Synchronous	SYSCLK / 12	8	None
1	Asynchronous	Timer 1 or 2 Overflow	8	1 Start, 1 Stop
2	Asynchronous	SYSCLK / 32 or SYSCLK / 64	9	1 Start, 1 Stop
3	Asynchronous	Timer 1 or 2 Overflow	9	1 Start, 1 Stop

Mise en œuvre de la liaison série asynchrone



Mise en œuvre de la liaison série asynchrone

Figure 22.3. T1 Mode 2 (8-bit Auto-Reload) Block Diagram



1. Timer Mode Register **TMOD**
2. Clock Control Register **CKCON**
3. Timer Control Register **TCON**
4. Timer Low Byte/High Byte **TH1, TL1**

Mise en œuvre de la liaison série asynchrone

Application pratique d'une configuration de timer :

Valeur de rechargement à mettre dans TH1 :

- On veut un débordement 153600 fois par seconde :
  - Durée entre les débordements ? $T_{souhaité}$ = …… µs
- On cadence le timer avec SYSCLK = 22,1184MHz
  - Durée entre les incrémentations du timer ? T_{timer} = …… ns
  - Combien d'incréments sont nécessaires entre les débordements ? ……
- Cela rentre-t-il dans la capacité du timer ? **Oui** – **Non**
  - **Si non : quoi faire ?**
- Valeur de rechargement ? ……

Mise en œuvre de la liaison série asynchrone

Figure 22.1. CKCON: Clock Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	T4M	T2M	T1M	T0M	Reserved	Reserved	Reserved	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0x8E

Bit7: UNUSED. Read = 0b, Write = don't care.

Bit6: T4M: Timer 4 Clock Select.

X

This bit controls the division of the system clock supplied to Timer 4. This bit is ignored when the timer is in baud rate generator mode or counter mode (i.e. C/T4 = 1).
0: Timer 4 uses the system clock divided by 12.
1: Timer 4 uses the system clock.

Bit5: T2M: Timer 2 Clock Select.

X

This bit controls the division of the system clock supplied to Timer 2. This bit is ignored when the timer is in baud rate generator mode or counter mode (i.e. C/T2 = 1).
0: Timer 2 uses the system clock divided by 12.
1: Timer 2 uses the system clock.

Bit4: T1M: Timer 1 Clock Select.

1

This bit controls the division of the system clock supplied to Timer 1.
0: Timer 1 uses the system clock divided by 12.
1: Timer 1 uses the system clock.

Bit3: T0M: Timer 0 Clock Select.

X

This bit controls the division of the system clock supplied to Counter/Timer 0.
0: Counter/Timer uses the system clock divided by 12.
1: Counter/Timer uses the system clock.

Bits2-0: Reserved. Read = 000b, Must Write = 000.

X : ne pas modifier
→ Utiliser des opérateurs binaires (masques)

Mise en œuvre de la liaison série asynchrone

Figure 22.5. TCON: Timer Control Register

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

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

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

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

Mise en œuvre de la liaison série asynchrone

Figure 22.6. TMOD: Timer Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0x89

Timer 1

Bit7: GATE1: Timer 1 Gate Control.

0

0: Timer 1 enabled when TR1 = 1 irrespective of /INT1 logic level.

1: Timer 1 enabled only when TR1 = 1 AND /INT1 = logic 1.

Bit6: C/T1: Counter/Timer 1 Select.

0

0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.4).

1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).

Bits5-4: T1M1-T1M0: Timer 1 Mode Select.

These bits select the Timer 1 operation mode.

1

0

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

Mise en œuvre de la liaison série asynchrone

Figure 20.8. SCON0: UART0 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SM00/FE0	SM10/RXOV0	SM20/TXCOL0	REN0	TB80	RB80	TI0	RI0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x98

Bits7-6: The function of these bits is determined by the SSTAT0 bit in register PCON.
If SSTAT0 is logic 1, these bits are UART0 status indicators as described in **Section 20.3**.
If SSTAT0 is logic 0, these bits select the Serial Port Operation Mode as shown below.
SM00-SM10: Serial Port Operation Mode:

01

SM00	SM10	Mode
0	0	Mode 0: Synchronous Mode
0	1	Mode 1: 8-Bit UART, Variable Baud Rate
1	0	Mode 2: 9-Bit UART, Fixed Baud Rate
1	1	Mode 3: 9-Bit UART, Variable Baud Rate

Bit5: SM20: Multiprocessor Communication Enable.
If SSTAT0 is logic 1, this bit is a UART0 status indicator as described in **Section 20.3**.
If SSTAT0 is logic 0, the function of this bit is dependent on the Serial Port Operation Mode.

1

Mode 0: No effect.
Mode 1: Checks for valid stop bit.
0: Logic level of stop bit is ignored.
1: RI0 will only be activated if stop bit is logic level 1.
Modes 2 and 3: Multiprocessor Communications Enable.
0: Logic level of ninth bit is ignored.
1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1 and the received address matches the UART0 address or the broadcast address.

Mise en œuvre de la liaison série asynchrone

Bit4:	REN0: Receive Enable. This bit enables/disables the UART0 receiver. 0: UART0 reception disabled. 1: UART0 reception enabled.
1	
Bit3:	TB80: Ninth Transmission Bit.
X	The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.
Bit2:	RB80: Ninth Receive Bit.
X	The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM20 is logic 0, RB80 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.
Bit1:	TI0: Transmit Interrupt Flag.
1!!	Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software
Bit0:	RI0: Receive Interrupt Flag.
0	Set by hardware when a byte of data has been received by UART0 (as selected by the SM20 bit). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.

Mise en œuvre de la liaison série asynchrone

Figure 22.14. T2CON: Timer 2 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF2	EXF2	RCLK0	TCLK0	EXEN2	TR2	C/T2	CP/RL2	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xC8

Bit7: TF2: Timer 2 Overflow Flag.
Set by hardware when Timer 2 overflows. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. TF2 will not be set when RCLK0 and/or TCLK0 are logic 1.

Bit6: EXF2: Timer 2 External Flag.
Set by hardware when either a capture or reload is caused by a high-to-low transition on the T2EX input pin and EXEN2 is logic 1. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 Interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit5: RCLK0: Receive Clock Flag for UART0.

0

Selects which timer is used for the UART0 receive clock in modes 1 or 3.
0: Timer 1 overflows used for receive clock.
1: Timer 2 overflows used for receive clock.

Mise en œuvre de la liaison série asynchrone

Figure 12.15. PCON: Power Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SMOD0	SSTAT0	Reserved	SMOD1	SSTAT1	Reserved	STOP	IDLE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x87

Bit7: SMOD0: UART0 Baud Rate Doubler Enable.
This bit enables/disables the divide-by-two function of the UART0 baud rate logic for configurations described in the UART0 section.
0: UART0 baud rate divide-by-two enabled.
1: UART0 baud rate divide-by-two disabled.

1

Bit6: SSTAT0: UART0 Enhanced Status Mode Select.
This bit controls the access mode of the SM20-SM00 bits in register SCON0.
0: Reads/writes of SM20-SM00 access the SM20-SM00 UART0 mode setting.
1: Reads/writes of SM20-SM00 access the Framing Error (FE0), RX Overrun (RXOV0), and TX Collision (TXCOL0) status bits.

Bit5: Reserved. Read is undefined. Must write 0.

Pour terminer : écrire dans SBUF0 pour déclencher l'envoi

Utilisation de bibliothèques de haut niveau (printf()…)

Le 8051 est un µC 8 bits. Rappel des caractéristiques :

- **Mémoire Flash (= code) : 64 Ko**
- **RAM : 4 Ko**

Code d'exemple utilisé SANS printf()

Utilisation RAM **127** octets, flash **933** octets

Code d'exemple utilisé AVEC printf()

Utilisation RAM **159** octets, flash **3546** octets

→ **Ne pas utiliser les fonctions de formatage de haut-niveau (printf, scanf et dérivés), ou au moins en connaissance de cause !** Il y a des alternatives plus légères (atoi(), itoa()…)

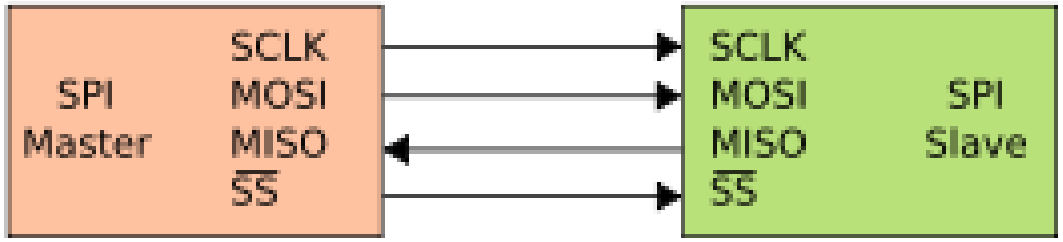
Deuxième séance de cours :

Liaison série synchrone

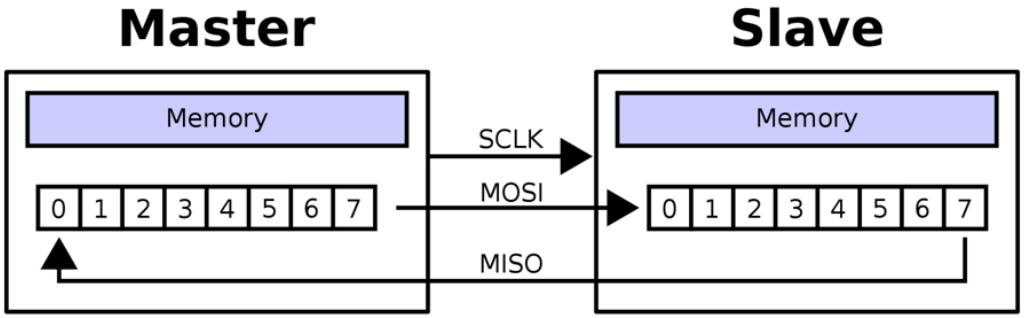
SPI

Bus synchrone : liaison SPI

Bus SPI : *Serial Peripheral Interface*



1. Communication full-duplex (dans les deux sens en même temps)
2. Liaison rapide : max SYSCLK/10 pour le 8051
3. Le 8051 peut être utilisé comme maître mais aussi comme esclave
4. Possibilité d'avoir plusieurs maîtres et/ou plusieurs esclaves
5. Seul le maître initialise le transfert : **il faut écrire pour pouvoir lire**
6. La liaison est basée sur un échange de données (2 registres à décalage)



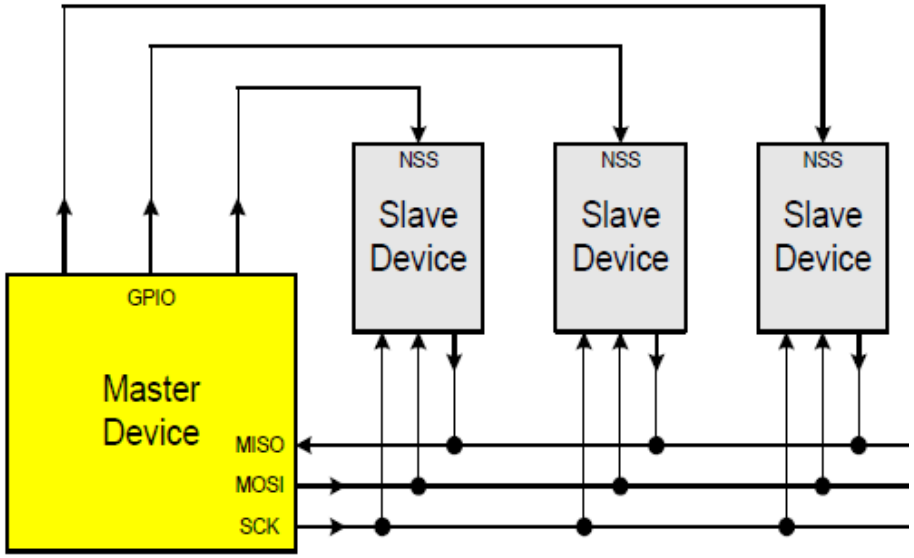
Bus synchrone : liaison SPI

Bus SPI : les signaux

- SCK (SPI Clock) : horloge
- MOSI : Master Out Slave In
  - sortie donnée maître, entrée donnée esclave
- MISO : Master In Slave Out
  - entrée donnée maître, sortie donnée esclave
- NSS : Slave Select
  - **Utilisé uniquement sur les esclaves**

Inconvénient : chaque esclave sur le bus requiert une I/O sur le maître pour y connecter son signal NSS.

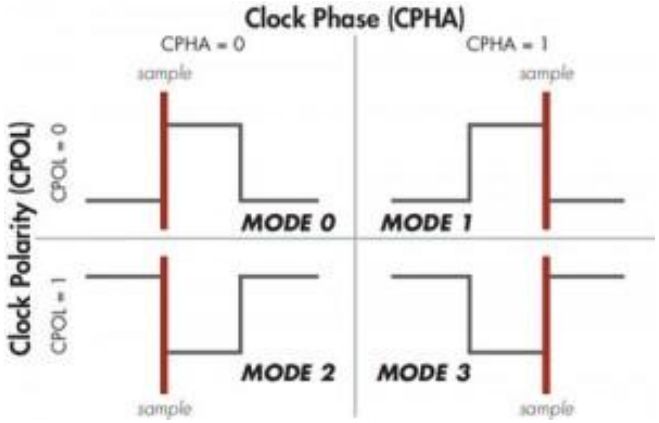
Le maître utilise des GPIO pour piloter le NSS des esclaves et non sa propre broche NSS !



Bus synchrone : liaison SPI

Bus SPI : horloge

- Quatre modes et trois paramètres :
  - La fréquence de l'horloge
  - La polarité de l'horloge CKPOL : Clock Polarity : niveau de repos haut ou bas
  - La phase de l'horloge CKPHA : Clock Phase : transmission sur front montant ou descendant
- Il faut les mêmes paramètres sur le maître et l'esclave
- La fréquence de l'horloge est fixée par le maître
- Les associations de CKPOL et CKPHA donnent donc 4 modes :



Bus synchrone : liaison SPI
Mise en œuvre sur le 8051

SPI0CFG

Registre de configuration :

- Phase
- Polarité
- Taille de la trame

Figure 19.5. SPI0CFG: SPI0 Configuration Register

R/W	R/W	R	R	R	R/W	R/W	R/W	Reset Value
CKPHA	CKPOL	BC2	BC1	BC0	SPIFRS2	SPIFRS1	SPIFRS0	00000111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9A

Bit7: CKPHA: SPI0 Clock Phase.
This bit controls the SPI0 clock phase.
0: Data sampled on first edge of SCK period.
1: Data sampled on second edge of SCK period.

Bit6: CKPOL: SPI0 Clock Polarity.
This bit controls the SPI0 clock polarity.
0: SCK line low in idle state.
1: SCK line high in idle state.

Bits5-3: BC2-BC0: SPI0 Bit Count.
Indicates which of the up to 8 bits of the SPI0 word have been transmitted.

BC2-BC0			BIT Transmitted
0	0	0	Bit 0 (LSB)
0	0	1	Bit 1
0	1	0	Bit 2
0	1	1	Bit 3
1	0	0	Bit 4
1	0	1	Bit 5
1	1	0	Bit 6
1	1	1	Bit 7 (MSB)

Bits2-0: SPIFRS2-SPIFRS0: SPI0 Frame Size.
These three bits determine the number of bits to shift in/out of the SPI0 shift register during a data transfer in master mode. They are ignored in slave mode.

SPIFRS			Bits Shifted
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

Bus synchrone : liaison SPI
Mise en œuvre sur le 8051

SPI0CN

Registre de contrôle :

- SPIF
- TXBUSY
- SLVSEL
- MSTEN
- SPIEN

Figure 19.6. SPI0CN: SPI0 Control Register

R/W	R/W	R/W	R/W	R	R	R/W	R/W	Reset Value
SPIF	WCOL	MODF	RXOVRN	TXBSY	SLVSEL	MSTEN	SPIEN	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (bit addressable)	SFR Address: 0xF8

Bit7: SPIF: SPI0 Interrupt Flag.
This bit is set to logic 1 by hardware at the end of a data transfer. If interrupts are enabled, setting this bit causes the CPU to vector to the SPI0 interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software.

Bit6: WCOL: Write Collision Flag.
This bit is set to logic 1 by hardware (and generates a SPI0 interrupt) to indicate a write to the SPI0 data register was attempted while a data transfer was in progress. If interrupts are enabled, setting this bit causes the CPU to vector to the SPI0 interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software.

Bit5: MODF: Mode Fault Flag.
This bit is set to logic 1 by hardware (and generates a SPI0 interrupt) when a master mode collision is detected (NSS is low and MSTEN = 1). If interrupts are enabled, setting this bit causes the CPU to vector to the SPI0 interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software.

Bit4: RXOVRN: Receive Overrun Flag.
This bit is set to logic 1 by hardware (and generates a SPI0 interrupt) when the receive buffer still holds unread data from a previous transfer and the last bit of the current transfer is shifted into the SPI0 shift register. If interrupts are enabled, setting this bit causes the CPU to vector to the SPI0 interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software.

Bit3: TXBSY: Transmit Busy Flag.
This bit is set to logic 1 by hardware while a master mode transfer is in progress. It is cleared by hardware at the end of the transfer.

Bit2: SLVSEL: Slave Selected Flag.
This bit is set to logic 1 whenever the NSS pin is low indicating it is enabled as a slave. It is cleared to logic 0 when NSS is high (slave disabled).

Bit1: MSTEN: Master Mode Enable.
0: Disable master mode. Operate in slave mode.
1: Enable master mode. Operate as a master.

Bit0: SPIEN: SPI0 Enable.
This bit enables/disables the SPI.
0: SPI disabled.

Bus synchrone : liaison SPI
Mise en œuvre sur le 8051

SPI0CKR

Registre de sélection de fréquence d'horloge

Compromis débit/distance à effectuer

Figure 19.7. SPI0CKR: SPI0 Clock Rate Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9D

Bits7-0: SCR7-SCR0: SPI0 Clock Rate
These bits determine the frequency of the SCK output when the SPI0 module is configured for master mode operation. The SCK clock frequency is a divided down version of the system clock, and is given in the following equation, where *SYSCLK* is the system clock frequency and *SPI0CR* is the 8-bit value held in the SPI0CR register.

$$f_{SCK} = \frac{SYSCLK}{2 \times (SPI0CKR + 1)}$$

for 0 <= SPI0CKR <= 255

Example: If SYSCLK = 2 MHz and SPI0CKR = 0x04,

$$f_{SCK} = \frac{2000000}{2 \times (4 + 1)}$$

$$f_{SCK} = 200kHz$$

Bus synchrone : liaison SPI

SPI0DAT : Registre de données.

Une écriture dans ce registre déclenche automatiquement son envoi sur la liaison.

A l'issue de la transmission, il contient la valeur du registre équivalent de l'esclave avant la transmission

Figure 19.8. SPI0DAT: SPI0 Data Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9B

Bits7-0: SPI0DAT: SPI0 Transmit and Receive Data.
The SPI0DAT register is used to transmit and receive SPI0 data. Writing data to SPI0DAT places the data immediately into the shift register and initiates a transfer when in Master Mode. A read of SPI0DAT returns the contents of the receive buffer.

Accéléromètre ADXL345

- Accéléromètre 3 axes, alimentation 3,3V
- Communication SPI 4 fils. Quel mode ?
- Lire dans la doc comment lire/écrire les registres
- Configuration des registres 0x31, 0x2C, 0x2D à faire
- Créer les fonctions :
  - Initialisation SPI
  - Transmission SPI
  - Lecture d'un registre
  - Ecriture d'un registre

Liaison série synchrone

I2C / SMBUS

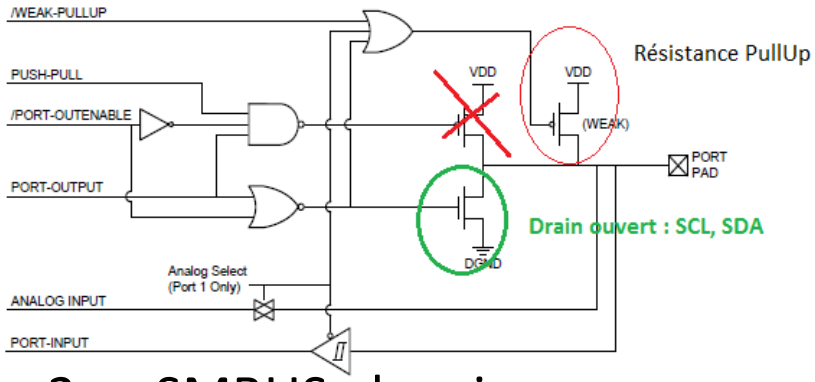
Bus série synchrone : I²C/SMBUS

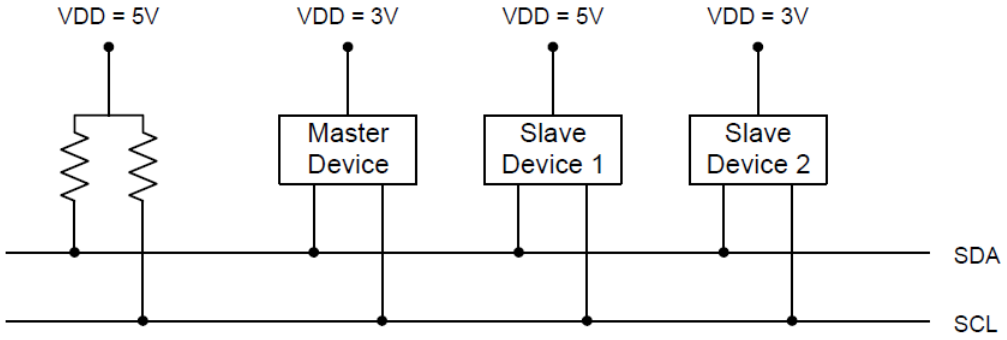
BUS I2C / SMBUS sur le microcontrôleur 8051

1. Plusieurs maîtres ou plusieurs esclaves
2. Bus bidirectionnel half-duplex
3. Si plusieurs maîtres démarrent la transmission en même temps, le plus lent ou celui qui écrit le plus de 0 (état dominant) garde la main
4. Celui qui envoie le **START** et l'**@** d'un esclave devient maître
5. Le maître fournit l'horloge sur le fil SCL
6. Chaque composant a une **adresse unique dont quelques bits sont souvent paramétrables, les autres étant les mêmes pour chaque référence de composant**
7. Sorties à drain ouvert sur SDA et SCL : bus à l'état haut quand il est libre
8. Liaison 100kbps en vitesse standard, 400kbps en rapide

Bus série synchrone : I²C/SMBUS

1. SMBUS : câblage physique

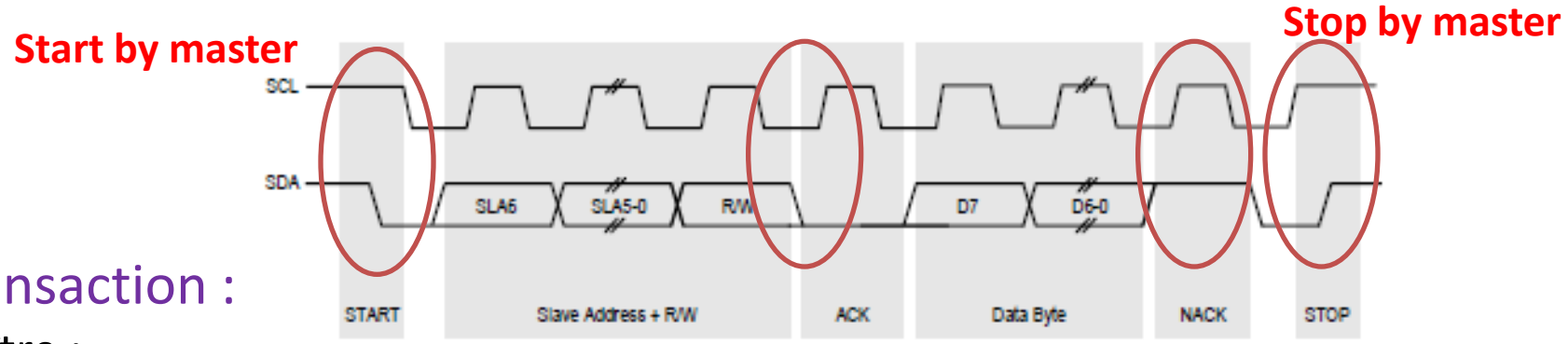




2. SMBUS : les signaux
   - Deux signaux : SDA (données) et SCL (horloge)
3. SMBUS : protocole
   - Le transfert de données se fait quand le bus est libre (état haut grâce à la résistance de pull-up)
   - Le bus est libre si SCL et SDA restent à l'état haut pendant plus **de 50µs**.
   - Si SCL est à l'état bas, aucune communication n'est possible, le maître ne peut pas forcer un état haut.

Bus série synchrone : I²C/SMBUS

4. SMBUS : protocole
   - La spécification complète est assez complexe



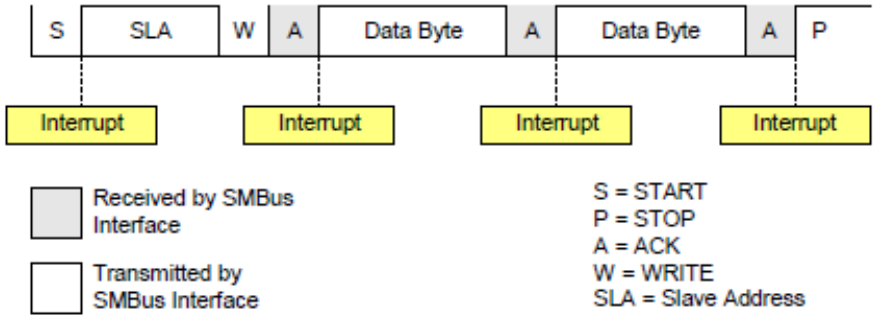
La transaction :

Le maître :

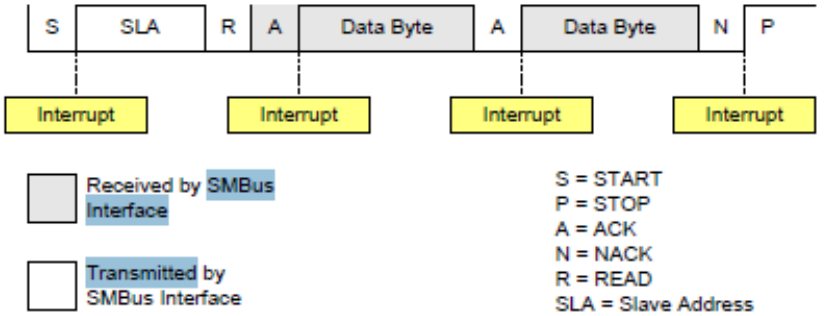
1. Démarre par une condition de **Start** (SDA tombe pendant que SCL reste haut)
2. Envoie l'adresse de l'esclave sur **7 bits + 1 bit (LSB) qui permet de le mettre en mode lecture ou écriture**
3. Transmission ou réception de un ou plusieurs octets de données (8 bits)
4. Chaque donnée reçue par le maître ou par l'esclave est suivie d'un acquiescement **(ACK)** ou non acquiescement **(NACK)**
5. À la fin, une condition de **STOP** est générée par le maître (SDA remonte pendant que SCL est à l'état haut)

Bus série synchrone : I²C/SMBUS

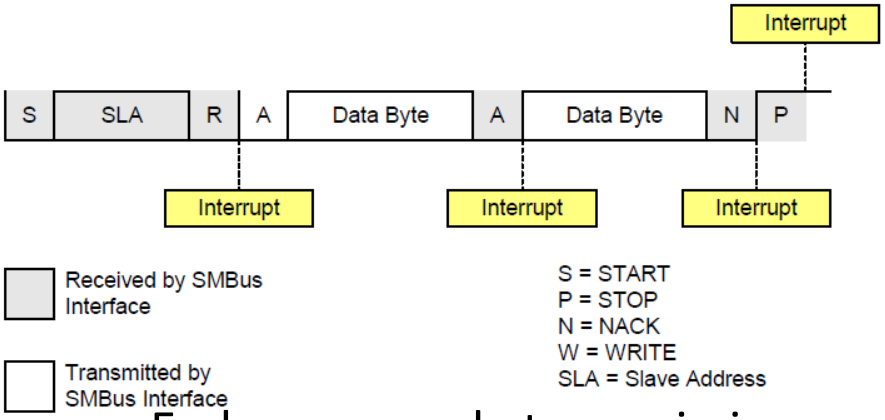
SMBUS : 4 mode de transfert



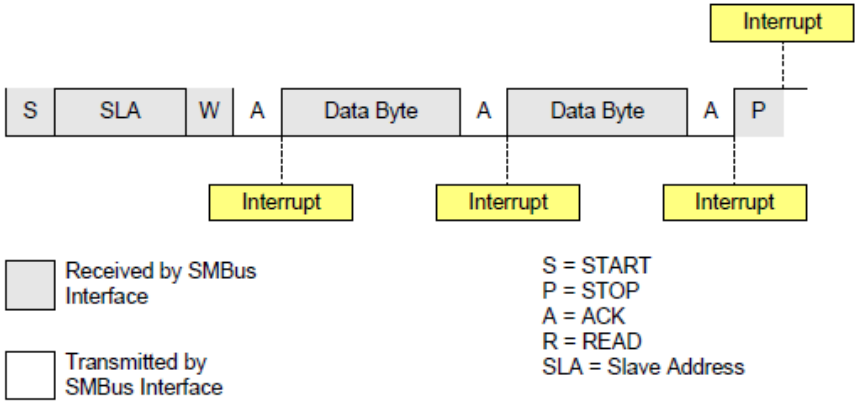
Maître en mode transmission



Maître en mode réception



Esclave en mode transmission



Esclave en mode réception